21 世纪高等职业教育计算机系列规划教材

JSP 动态网站开发项目教程

徐婉珍　主　编

刘　超　杨立超　副主编

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书以基于典型工作任务的课程开发为原则来组织本课程的内容。书中配备了大量的例题及系统实例，叙述详细，通俗易懂，书中所有程序代码均在开发环境中调试通过。

本书分为 4 个部分，共 20 章。第一部分为开发前的知识与技能准备，介绍了开发任务、JSP 的基本概念，以及主流开发工具 NetBeans IDE 的安装与使用；第二部分进入开发与学习阶段，以 NetBeans IDE 为开发工具，介绍论坛网站主要模块的开发过程，在开发的过程中穿插讲解了 JSP 的语言基础、JSP 的页面元素、内置对象、JDBC 访问数据库、JavaBean 的使用、Servlet、文件上传、异常处理、服务器的配置与网站的发布等知识；第三部分以 MyEclipse 为开发工具，提供一个实现了 MVC 模式的企业网站实例，展示了 Servlet+JSP+JavaBeans 技术开发动态网站的基本流程，同时介绍了 JSP 的 EL、JSTL、过滤器等相关知识；第四部分为 JSP 速查参考，系统地概括了 JSP 技术各部分的知识，最后对主流开发工具 MyEclipse 的安装和使用进行了详细的介绍。

本书可作为普通高等院校 JSP 程序设计课程的教材，也可作为初学者的自学用书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

JSP 动态网站开发项目教程 / 徐婉珍主编. —北京：电子工业出版社，2010.11
（21 世纪高等职业教育计算机系列规划教材）
ISBN 978-7-121-12168-5

Ⅰ. ①J… Ⅱ. ①徐… Ⅲ. ①JAVA 语言－主页制作－程序设计－高等学校：技术学校－教材 Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字（2010）第 214555 号

策划编辑：徐建军
责任编辑：徐　磊
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：19.25　字数：492.8 千字
印　　次：2010 年 11 月第 1 次印刷
印　　数：3 000 册　　定价：30.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

随着 Internet 的普及和发展，越来越多的企业级应用采用了 B/S 的模式，即以动态网站的形式向用户提供软件服务。1999 年 Sun 公司推出了 JSP 动态网页技术，由于采用 JSP 技术开发的系统具有跨平台、安全性好、效率高等优势，因此 JSP 被广泛地应用于 Web 应用系统的软件开发中，在 Internet 上基于 Java Web 技术的动态网站到处可见。为满足社会对软件人才的需求，各大院校也大都为计算机软件类专业的学生开设了类似的课程。本书旨在为学习 JSP 技术的读者提供一种全新而又自然的学习模式。

■ 本书主要特色

为了使读者能快速地掌握 JSP 技术，本书遵循人的自然认知规律，按照“任务驱动，按需学习”的思想设计了新颖的整体结构。主体部分以论坛网站几大功能模块的开发为线索，根据任务开发中的技术需求来介绍相关的 JSP 知识，即“需要懂什么，就学习什么”。

采用这种学习模式，一方面可在短时间内看到学习的成果，有助于培养学习的兴趣，引导学习进入良性循环的轨道；另一方面，可为读者构建一个实际的应用场景，加深读者对各项技术的理解，充分了解其使用场合及使用方法。

书中还详细介绍了两大主流开发工具 NetBeans IDE 和 MyEclipse 的安装与使用方法。

■ 本书的主要内容

本书按照 Java Web 开发程序员的岗位能力要求选择内容，全书分为 4 个部分。

第一部分介绍任务开发前的准备知识，具体内容如下。

第 1 章　开发任务描述，介绍作为学习案例的论坛项目的基本需求。

第 2 章　JSP 概述，讲解 JSP 的基本概念。

第 3 章　开发工具的安装与使用，详细介绍 NetBeans IDE 的安装与使用方法。

第二部分为本书的核心部分，介绍了在 NetBeans 中开发论坛网站的过程，涵盖了 JSP 的指令与动作、内建对象、数据库访问、JavaBean 的使用、Servlet 初步，以及文件上传、在线编辑器的使用等实用技术，学习的过程将伴随着模块功能的完成而展开，各章具体内容如下。

第 4 章　论坛网站的设计，描述了论坛的功能模块设计、页面设计及数据库结构设计。

第 5 章　制作静态网页，介绍了利用 HTML、CSS 及 JavaScript 制作静态网页的基础知识。

第 6 章　注册功能的实现，讲解注册功能的实现过程及其中涉及的相关知识点。

第 7 章　登录功能的实现，讲解登录功能的实现过程及其中涉及的相关知识点。

第 8 章　帖子浏览功能的实现，讲解帖子列表显示功能的实现过程及其中涉及的相关知识点。

第 9 章　回帖功能的实现，讲解回帖功能的实现过程及其中涉及的相关知识点。

第 10 章 功能的增强与改进，在改进注册功能的过程中，介绍了较高级的 JSP 技术，如 Servlet、文件上传、Ajax 技术和在线编辑器的使用等。

为了给读者提供综合性更强的学习案例，第三部分详细讲解了一个企业网站的开发过程，向读者展示了如何使用 Servlet+JSP+JavaBean 实现 MVC 模式的网站系统，案例中还使用了过滤器、EL 及 JSTL 等高级技术。该案例使用了 MyEclipse 作为开发工具。

第四部分为 JSP 速查参考，目的是为了方便读者查找相关的知识，同时也能帮助读者构建相对完整的知识结构，最后介绍了主流开发工具 MyEclipse 的安装与使用方法。

读者在阅读书中源代码时需注意，书中所标注的行号并非程序源码中的真实行号，仅为便

于讲解而重新设定的序号。

■ 适合的读者对象

本书采取了在开发中学习的模式，由浅入深地讲解了 JSP 各项知识，所选取的内容切合实际开发的需要，力求使读者在最短的学习时间内掌握 JSP 技术。第一、二部分在每章后针对难点、重点精心设计了相关的思考题与实训任务，可帮助读者进一步巩固相关学习内容。本书学习起点较低，讲解由浅入深，内容难度适中，可作为普通高等院校 JSP 程序设计课程的教材，也可作为自学者的学习用书。

本书由徐婉珍任主编，刘超、杨立超任副主编，李强、魏菊霞、于枫参与了编写工作。

为了方便教师教学，本书配有电子教学课件及案例源代码，请有此需要的教师登录华信教育资源网（www.hxedu.com.cn）免费注册后进行下载，有问题时请在网站留言板留言或与电子工业出版社联系（E-mail:hxedu@phei.com.cn），也可与作者联系（E-mail: jspbook@126.com）。

由于对项目式教学法正处于经验积累和改进的过程中，同时，由于编者水平有限和时间仓促，书中难免存在疏漏和不足。希望同行专家和读者能给予批评和指正。

编　者

目　录

第一部分　预备阶段

第 1 章　开发任务描述 …… (1)

1.1　开发背景 …… (1)

1.2　需求说明 …… (1)

1.2.1　术语 …… (1)

1.2.2　功能需求 …… (2)

1.2.3　系统用例 …… (2)

思考与训练 …… (4)

第 2 章　JSP 概述 …… (5)

2.1　静态网页与动态网页 …… (5)

2.2　主流动态网站技术及实现模式 …… (5)

2.2.1　几种主流的动态网站开发技术 …… (5)

2.2.2　动态网站技术的实现模式 …… (6)

2.3　HTTP 的请求/响应模型 …… (7)

2.4　JSP 页面的执行流程 …… (9)

思考与训练 …… (9)

第 3 章　开发工具的安装与使用 …… (10)

3.1　NetBeans IDE 的安装 …… (10)

3.1.1　下载安装包 …… (10)

3.1.2　安装 NetBeans IDE …… (10)

3.2　创建项目与编写程序 …… (13)

3.2.1　认识 NetBeans IDE 的界面 …… (14)

3.2.2　创建新项目 …… (16)

3.2.3　修改默认模板 …… (19)

3.2.4　创建 JSP 页面 …… (20)

3.2.5　运行 JSP 页面 …… (22)

3.2.6　认识 JSP 的页面元素 …… (22)

3.2.7　添加自定义的模板 …… (23)

3.2.8　打开/关闭项目 …… (25)

3.3　在 NetBeans IDE 中调试程序 …… (25)

思考与训练 …… (28)

第二部分　在开发中学习

第 4 章　论坛网站的设计 …… （29）

4.1　功能模块设计 …… （30）

4.2　页面列表 …… （31）

4.3　数据库结构设计 …… （31）

思考与训练 …… （33）

第 5 章　制作静态网页 …… （34）

5.1　制作网站的原型 …… （34）

5.2　制作注册页面 …… （35）

5.2.1　HTML 的常用标签 …… （36）

5.2.2　CSS 简介 …… （41）

5.2.3　使用 HTML＋CSS 制作注册页面 …… （46）

5.2.4　JavaScript 简介 …… （49）

5.2.5　为注册表单添加 JavaScript 验证 …… （52）

思考与训练 …… （59）

第 6 章　注册功能的实现 …… （61）

6.1　任务的提出与分解 …… （61）

6.2　输出信息及提取表单数据 …… （62）

6.2.1　使用 page 指令设置页面编码 …… （62）

6.2.2　使用 out 对象输出信息 …… （62）

6.2.3　使用 request 对象读取请求参数 …… （63）

6.2.4　任务实现 …… （65）

6.2.5　重点提示 …… （66）

6.2.6　拓展与提高 …… （66）

6.3　验证用户名的唯一性（JDBC-ODBC 驱动） …… （69）

6.3.1　使用 page 指令导入类包 …… （70）

6.3.2　查询数据库的一般过程 …… （70）

6.3.3　实验案例 …… （72）

6.3.4　任务实现 …… （75）

6.3.5　重点提示 …… （77）

6.3.6　拓展与提高 …… （77）

6.4　将注册信息插入到用户信息表中（一） …… （78）

6.4.1　任务实现 …… （79）

6.4.2　重点提示 …… （81）

6.5　将注册信息插入到用户信息表中（二） …… （81）

6.5.1　PreparedStatement 接口 …… （81）

6.5.2　实验案例 …… （83）

6.5.3　任务实现 …… （84）

6.5.4 重点提示 …… （84）
6.6 页面跳转控制 …… （84）
6.6.1 <jsp:forward>动作 …… （85）
6.6.2 response 对象的重定向方法 …… （87）
6.6.3 任务实现 …… （87）
6.6.4 重点提示 …… （89）
思考与训练 …… （89）
第 7 章 登录功能的实现 …… （91）
7.1 任务的提出与分解 …… （91）
7.2 验证登录信息（JDBC 驱动） …… （91）
7.2.1 通过 JDBC 访问数据库 …… （91）
7.2.2 任务实现 …… （93）
7.2.3 重点提示 …… （95）
7.3 在会话期间维持用户的登录信息 …… （95）
7.3.1 session 对象的工作机制 …… （95）
7.3.2 实验案例 …… （96）
7.3.3 任务实现 …… （98）
7.3.4 重点提示 …… （99）
7.4 公共页面的处理 …… （99）
7.4.1 include 指令 …… （100）
7.4.2 <jsp:include>动作 …… （101）
7.4.3 任务实现 …… （102）
7.4.4 重点提示 …… （103）
7.5 在线会员名单的显示 …… （103）
7.5.1 application 对象的使用 …… （103）
7.5.2 任务实现 …… （105）
7.5.3 重点提示 …… （108）
7.6 使用 Cookie 完成自动登录功能 …… （108）
7.6.1 任务提出 …… （108）
7.6.2 Cookie 的使用 …… （109）
7.6.3 任务实现 …… （110）
7.6.4 重点提示 …… （112）
思考与训练 …… （112）
第 8 章 帖子浏览功能的实现 …… （114）
8.1 显示所有帖子的概要信息 …… （114）
8.1.1 任务提出 …… （114）
8.1.2 JSP 的表达式 …… （114）
8.1.3 任务实现 …… （114）
8.1.4 重点提示 …… （118）

8.2 搜索帖子 …… （118）
8.2.1 任务提出 …… （118）
8.2.2 自提交的表单 …… （118）
8.2.3 任务实现 …… （119）
8.3 分页显示 …… （121）
8.3.1 任务提出 …… （121）
8.3.2 分页显示的原理 …… （121）
8.3.3 任务实现 …… （122）
8.3.4 重点提示 …… （125）
思考与训练 …… （125）
第 9 章 回帖功能的实现 …… （126）
9.1 任务提出 …… （126）
9.2 输入回复帖的页面 …… （126）
9.2.1 HTML 标签——textarea …… （126）
9.2.2 在线编辑器 FCKeditor 的使用 …… （127）
9.2.3 任务实现 …… （129）
9.2.4 重点提示 …… （134）
思考与训练 …… （134）
第 10 章 功能的增强与改进 …… （135）
10.1 数据库通用处理类 …… （135）
10.1.1 任务提出 …… （135）
10.1.2 编写通用处理类 …… （135）
10.1.3 使用通用处理类 …… （137）
10.2 JavaBean 的使用 …… （139）
10.2.1 任务提出 …… （139）
10.2.2 JavaBean 的基本概念 …… （139）
10.2.3 通过 JSP 动作使用 JavaBean …… （140）
10.2.4 JavaBean 的使用案例 …… （141）
10.2.5 任务实现 …… （145）
10.2.6 拓展与提高 …… （148）
10.2.7 重点提示 …… （149）
10.3 使用 Servlet 完成用户注册功能 …… （150）
10.3.1 任务提出 …… （150）
10.3.2 Servlet 简介 …… （150）
10.3.3 使用 Servlet 接收表单数据 …… （153）
10.3.4 任务实现 …… （156）
10.3.5 重点提示 …… （158）
10.4 使用 Ajax 实现用户名唯一性的即时检测 …… （158）
10.4.1 任务提出 …… （158）

10.4.2 Ajax 简介 …… （158）
10.4.3 Ajax 的基本工作原理 …… （158）
10.4.4 任务实现 …… （158）
10.5 为用户注册添加头像上传功能 …… （163）
10.5.1 任务提出 …… （163）
10.5.2 jspSmartUpload 的相关类 …… （163）
10.5.3 使用 jspSmartUpload 上传文件 …… （166）
10.5.4 任务实现 …… （170）
10.5.5 重点提示 …… （174）
10.6 为注册输入页面添加验证码 …… （174）
思考与训练 …… （176）
第 11 章 服务器的配置与网站的发布 …… （178）
11.1 Tomcat 的安装与配置 …… （178）
11.1.1 Tomcat 的下载 …… （178）
11.1.2 Tomcat 的安装 …… （178）
11.1.3 Tomcat 的启动与停止 …… （180）
11.1.4 测试 Tomcat …… （180）
11.2 Tomcat 的目录结构 …… （181）
11.3 配置网站的异常处理 …… （181）
11.4 手动发布网站 …… （183）
11.5 利用 NetBeans 发布网站 …… （184）
思考与训练 …… （185）

第三部分 综合案例——企业网站

第 12 章 需求分析 …… （186）
12.1 开发背景 …… （186）
12.2 需求说明 …… （186）
12.2.1 用户角色 …… （187）
12.2.2 系统功能概述 …… （187）
12.3 可行性分析 …… （188）
第 13 章 系统设计 …… （189）
13.1 系统功能结构 …… （189）
13.1.1 网站的页面级别关系 …… （189）
13.1.2 网站的功能结构 …… （189）
13.1.3 系统架构 …… （191）
13.2 开发工具及运行环境 …… （192）
13.2.1 使用技术 …… （192）
13.2.2 开发工具 …… （192）
13.2.3 运行环境 …… （192）

13.3 项目的文件夹组织结构 …… (192)
13.4 系统预览 …… (193)
13.4.1 前台页面 …… (193)
13.4.2 后台页面 …… (195)
第 14 章 数据库设计 …… (199)
第 15 章 功能模块的设计与实现 …… (202)
15.1 公共类 …… (202)
15.1.1 数据库连接类 …… (202)
15.1.2 通用处理工具类 …… (205)
15.1.3 编码转换过滤器 …… (207)
15.1.4 数据库通用处理 …… (209)
15.1.5 公共页面 …… (211)
15.2 Model 层的设计 …… (211)
15.2.1 实体类设计 …… (211)
15.2.2 DAO 接口 …… (211)
15.3 前台功能模块 …… (214)
15.3.1 首页显示模块 …… (214)
15.3.2 文章显示模块 …… (220)
15.3.3 在线留言模块 …… (225)
15.4 后台功能模块 …… (229)
15.4.1 登录模块 …… (229)
15.4.2 文章管理模块 …… (233)
15.4.3 留言管理模块 …… (241)

第四部分 JSP 速查参考

第 16 章 JSP 的语言基础 …… (250)
16.1 HTML 语言基础 …… (250)
16.1.1 HTML 的标签和属性 …… (250)
16.1.2 HTML 页面的基本结构 …… (251)
16.1.3 HTML 的常用标签 …… (251)
16.2 JavaScript 简介 …… (256)
16.2.1 JavaScript 的基本语法 …… (256)
16.2.2 在页面中加入 JS 代码 …… (256)
16.2.3 JS 的事件处理 …… (256)
16.2.4 表单数据验证的一般方式 …… (257)
16.3 Java 语言基础 …… (257)
16.3.1 Java 的类和对象 …… (257)
16.3.2 常用的 Java 类 …… (258)
16.3.3 Java 的异常处理机制 …… (259)

第 17 章　JSP 的页面元素 ……（261）
17.1　JSP 的注释 ……（261）
17.2　JSP 的指令 ……（262）
17.2.1　page（页面）指令 ……（262）
17.2.2　include（包含）指令 ……（263）
17.2.3　taglib（标签库）指令 ……（263）
17.3　JSP 的代码段 ……（263）
17.4　JSP 的表达式 ……（264）
17.5　JSP 的标准动作 ……（264）
17.5.1　<jsp:param>动作 ……（264）
17.5.2　<jsp:include>动作 ……（265）
17.5.3　<jsp:forward>动作 ……（265）
17.5.4　使用 JavaBean 的系列动作 ……（265）
第 18 章　JSP 的内建对象 ……（267）
18.1　out 对象 ……（267）
18.2　request 对象 ……（268）
18.3　response 对象 ……（268）
18.4　session 对象 ……（269）
18.5　application 对象 ……（270）
18.6　config 对象 ……（270）
18.7　exception 对象 ……（270）
18.8　page 对象 ……（270）
18.9　pageContext 对象 ……（271）
第 19 章　JDBC 访问数据库 ……（272）
19.1　一般步骤 ……（272）
19.2　常用的接口或类 ……（274）
19.2.1　DriverManager 类 ……（274）
19.2.2　Connection 接口 ……（274）
19.2.3　Statement 接口 ……（274）
19.2.4　PreparedStatement 接口 ……（275）
19.2.5　CallableStatement 接口 ……（275）
19.2.6　ResultSet 接口 ……（276）
第 20 章　拾遗补缺 ……（277）
20.1　关系数据库与 SQL 回顾 ……（277）
20.1.1　关系数据库的基本结构 ……（277）
20.1.2　SQL 语言 ……（277）
20.2　MyEclipse 的安装与使用 ……（278）
20.2.1　Eclipse 及 MyEclipse 的安装 ……（278）
20.2.2　MyEclipse 的使用 ……（283）
20.3　参考链接 ……（293）

第一部分　预备阶段

下面将以论坛网站的开发为主线展开学习。本部分将对所要开发的任务进行介绍，讲述 JSP 的基本概念与原理，并详细介绍开发工具 NetBeans IDE 的安装过程，以及基本的使用方法。具体章节如下。

第 1 章　开发任务描述

第 2 章　JSP 概述

第 3 章　开发工具的安装与使用

第 1 章　开发任务描述

本书以网上论坛的开发过程为线索展开 JSP 技术的学习之旅。那么，我们究竟要完成一个怎样的论坛网站呢？下面对该系统的需求进行简单的介绍。

1.1　开 发 背 景

随着 Internet 的发展，网上论坛成为了当今网络世界非常普遍的交流平台。通过这一虚拟的网络平台，来自世界各地的人们得以突破时间与空间的限制，在此发布自己关心的信息，使信息的交流更加广泛、便捷；人们还可以参与各种话题，发表自己的意见，各种不同的观点在这里交汇碰撞，从而使问题的讨论更加深入、全面。可以说，网上论坛已经成为了人们现代生活的一部分，许多交流性网站都开辟了论坛版块，如各个城市的信息港论坛、各种专业技术论坛、企业或机构内部的交流论坛等。

无论创设论坛的目的是什么，网上论坛的功能基本都是相同的，即以提供信息的发布与浏览为主。这里以一个一般性的校园论坛网站作为开发目标，论坛的用户以校内师生为主，完成后可作为师生们学习、生活的交流平台，也可进一步扩展作为其他用途。下面对该论坛的需求进行简单的说明。

1.2　需 求 说 明

1.2.1　术语

为了使后面的描述更加清晰，首先对几个相关的术语进行定义。

1．用户

特指在网站中填写了注册资料并成功注册的论坛使用者，具有浏览、发帖等权限。

2．主题帖

注册用户发表的新帖子，通常会展开一个新的话题。发表主题帖的行为简称为“发帖”。

3．回复帖

指针对其他用户发布的帖子，发表自己的回复内容的帖子。发表回复帖的行为简称为“回帖”。

4．版块

为了使用户能快速找到自己感兴趣的内容，一般可将帖子分类管理。这里的“版块”即帖子所属的类别。版块只能由管理员设立或删除。

5．版主

具有特殊权限的注册用户，可对某一版块的帖子进行管理。由网站管理员进行指定或撤销。

6．管理员

具有后台操作权限的网站用户，一般为网站的工作人员，具有对网站进行系统管理的权限，如帖子管理、版块管理和公告管理等。

1.2.2　功能需求

网上论坛是为网民们提供的讨论与交流的场所，“听”和“说”是用户最常用的两项操作。“听”，即浏览他人发表的帖子；“说”，即参与到论坛中去发帖。因此，浏览帖子、发表主题帖和回复帖是论坛网站应具备的基本操作。

此外，为了记录发言者的基本信息，需要发言人注册为网站的会员，以便对其进行必要的管理，同时也可为会员提供更为个性化的服务。因此，注册功能是论坛网站必不可少的基本功能，注册用户必须在登录后才能发表主题帖或回复帖。

如果需要根据帖子的种类分设版块，一般还应为不同的版块指定版主，以便对用户发表的帖子进行管理，如执行删除、屏蔽、加精、置顶等操作。一般用户经网站管理员指定后，可成为某版块的版主。

管理员可对整个论坛的帖子进行管理，即版主具有的权限。

管理员具有版块管理的权限，可添加版块、删除版块或修改版块的名称。

网站可能要实时更新一些公告信息，因此，管理员应可对公告进行添加、修改和删除等。

除此以外，网站还可能有广告位的出租业务，管理员应可对此类信息进行管理。

最后，管理员应可对用户进行管理，如对论坛用户执行积分奖惩操作等。

1.2.3　系统用例

从前面的分析可以得知，论坛应有 3 类用户，即普通用户、版主和管理员，普通用户被管理员指定后可成为某个或某几个版块的版主。总结前面的主要功能需求，得到如图 1-1 所示的系统一级用例图。

对每个一级用例进行具体的分析，可以得到更为详细的二级用例图。例如，前台操作用例可进一步细化为如图 1-2 所示的二级用例图。

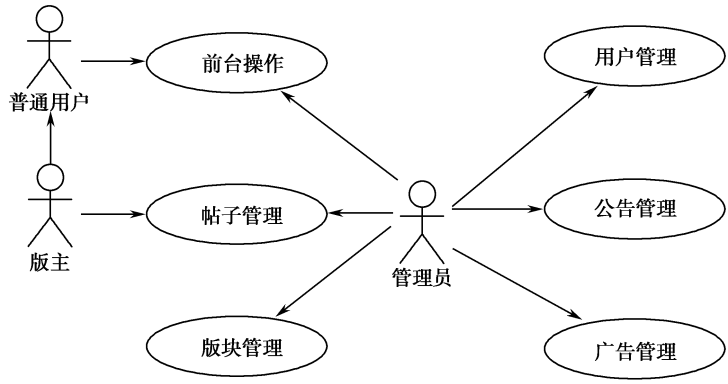


图 1-1　论坛系统一级用例图

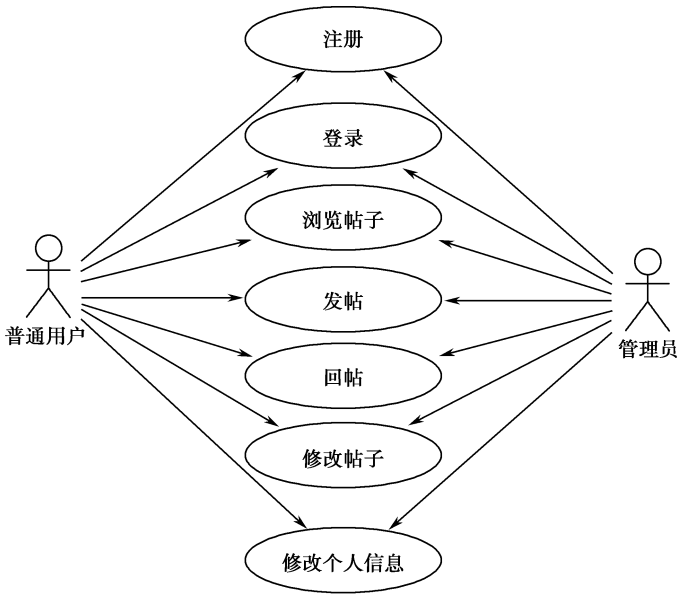


图 1-2　前台操作用例图

要开发一个功能全面的论坛，其业务逻辑还是比较复杂的。为了避免因业务逻辑的复杂性干扰了读者对技术本身的理解和掌握，后面章节将仅以开发单版块论坛的前台子系统作为学习案例，即图 1-2 所示的用例，包括普通用户的注册、登录、浏览帖子、发帖、回帖、修改帖子和个人信息等功能的实现。

前台操作用例中各个子用例的详细信息如下。

1．注册

新用户可填写自己的用户名、密码、E-mail、头像等个人信息，用户名是用户的唯一身份标识，在整个论坛系统中不可重复。为了表示用户在论坛中的参与度，还可为用户设立积分或等级信息。

2．登录

完成注册后，用户可输入用户名和密码登录论坛，登录后的用户具有看帖、发帖、回帖及修改旧帖等权限。

3．浏览帖子

登录用户可浏览论坛的帖子信息，包括主题帖的作者、标题、内容、发布时间、回复数、点击数，以及相关的回复帖等信息。为了衡量一个帖子的热门度，可为主题帖统计点击数和回复数。

4．发帖

登录用户可以发表新的话题，应填写主题帖的标题和内容。

5．回帖

登录用户在浏览主题帖的同时，可以针对该帖发表自己的回复内容。主题帖的回复数即相关回复帖的数量。

6．修改帖子

登录用户可以修改自己过去发表的主题帖或回复帖，包括标题和内容。

7．修改个人信息

注册后的用户可以修改自己的密码及头像等个人信息。设置新密码前应输入原来的旧密码。一般来说，注册后用户名和 E-mail 不可更改。

这里不再详细描述其他用例的信息，读者在学习和掌握了相关的技术之后，可自行扩展论坛其他的功能模块。



思考与训练

一、思考题

1．为提高用户参与发帖或回帖的积极性，可以为论坛增加哪些功能？

2．若论坛的用户之间希望能有一对一的短消息交流，则应该添加哪些功能？

3．应为论坛的管理员设立哪些功能？请分析并给出相关的用例图。

第 2 章　JSP 概述

JSP 的全称为 Java Server Pages，顾名思义，是用 Java 编写的运行在服务器端的网页。实际上，JSP 是由 Sun Microsystem 公司（现已被 Oracle 收购）倡导的、由许多公司参与建立的一种动态网页技术标准，是多种技术的综合运用。在传统的 HTML 网页文件中加入 Java 代码片段（Scriplet）和 JSP 自身特有的标记，就形成了 JSP 的动态网页。

2.1　静态网页与动态网页

在 Internet 发展的初期，一般网页主要是以信息发布为主的，不同的用户访问同一网页时，看到的内容都是相同的，如同我们订阅的日报，所有读者看到的报纸内容都无一例外。这时的网页不具有与用户交互的能力，不能针对不同的需求给予不同的反馈信息，在开发人员制作完毕后，其内容就几乎固定不变了，这样的网页我们称之为“静态网页”。

Internet 发展到今天，绝大多数的网页都不再是静态的了，它们实质上是以网页的形式提供给用户使用的应用程序，具有逻辑处理的能力。它们能接收用户的信息，并且可根据用户的需求给予相应的反馈，即具有与用户交互的功能。这些网页的本质是一些能处理用户业务需求的应用程序，或者说是以网页的形式承载的应用程序。

例如，网上银行，我们可以在其网站上实现查询账户、转账、付费等功能；又如，航空公司的票务网站，可以为客户提供航班查询、订票等业务。对于这些具备逻辑处理功能的网页来说，由于来自客户端的需求可能不同，即使不同的用户访问的是同一个网页，执行的代码是相同的，但是处理的结果却可能是不同的，因此呈现给客户端的网页内容也可能是不同的。例如，查询航班时，虽然使用的是同一个查询网页，但是用户可以查询不同的航班，或者同一个用户在不同的时候查询，该网页给出的查询结果都可能是不同的。

从技术的角度来说，传送给客户端的网页内容是在客户端发起请求后，由服务器执行相关的程序后动态生成的。因此这样的网页被称为“动态网页”，包含动态网页的网站自然就被称为“动态网站”了。

显然，网上论坛属于动态网站，因为论坛上所显示的帖子和回复都来自访问论坛的会员们，论坛网站必须具有与用户交互的能力，而网站所显示的内容也可能时刻不同。

那么，目前主流的动态网站开发技术有哪些呢？下面将对此进行简单的介绍。

2.2　主流动态网站技术及实现模式

2.2.1　几种主流的动态网站开发技术

1．ASP（Active Server Pages）

ASP 是由微软公司推出的动态网站开发技术，它使用 VBScript 或 JavaScript 作为脚本语言，基本上只能运行在 Windows 平台上。国内 ASP 的应用较广，与国际上 PHP 占主导地位的情况正好相反。不过近年来这种形势发生了改变，在小型网站的开发领域，PHP 逐渐占据了重要的位置。

2．PHP（Personal Home Pages）

PHP 使用的脚本语言类似于 C、Java、Perl 等语言，是免费使用的开源技术。开发者使用 PHP 技术可以快速地建立一个交互式的 Web 站点。但由于 PHP 本身的一些缺点，使得它不适合开发大型的电子商务站点，多用于创建比较小型的网站。

3．JSP（Java Server Pages）

JSP 是由 Sun 公司在 1999 年推出的动态网站开发技术。

在推出 JSP 之前，Sun 公司使用 Servlet 技术来开发网络应用，Servlet 一般是实现了 javax.servlet.Servlet 接口的 Java 类，可以处理用户请求并向客户端发回响应。使用 Servlet 技术开发一个网页时，无论是业务逻辑的处理还是页面效果的呈现，均使用 Java 语言编写代码实现，因此对程序开发人员的要求比较高，也在一定程度上阻碍了它的推广。

为了降低动态网站开发的难度，1999 年 6 月 Sun 公司在 Servlet 的基础上推出了 JSP 技术，主要是在 HTML 格式的页面代码中嵌入具有逻辑处理能力的 Java 代码。采用这种形式，使得程序员的分工成为可能，网页效果与逻辑处理可分别交由不同的开发人员完成。

由于使用了 Java 语言作为网页的脚本语言，使用 JSP 技术编写的网页具有跨平台的特性。

4．ASP.NET

ASP.NET 技术是微软公司推出的动态网页开发技术，它是可以使用在.NET CLR（Common Language Runtime）上的编程框架，可用于在服务器上生成功能强大的 Web 应用程序。它主要采用 VB 或 C#作为脚本语言。

2.2.2　动态网站技术的实现模式

静态网页一般使用 HTML 语言结合 CSS 及 JavaScript 实现，浏览器软件通常都内置了对这些语言的支持，因此能直接解释静态网页的内容并将结果显示给用户。但是，HTML 语言仅仅是一种描述页面静态元素的标签语言，并不具有逻辑处理的功能。例如，处理订票业务时，仅使用 HTML 语言是无法判断是否有指定航班的机票可供订购的。因此，需要使用具有逻辑处理功能的某种程序设计语言来编写这些处理代码。

纵观目前流行的几种动态网站开发技术，它们都采用了相似的实现模式，即以静态网页（HTML+CSS+JavaScript）为基础，在其中嵌入采用某种高级语言编写的逻辑代码段，使之具备“动态”生成网页内容的能力。

例如，ASP 技术就采用了在 HTML 格式的网页代码中嵌入用 VBScript 或 JavaScript 编写的代码段的形式，而 PHP 使用的则是类似于 Perl 的脚本语言。

JSP 也采用了上述的模式，它通过在传统的 HTML 页面中加入 Java 代码段及 JSP 标记来完成逻辑处理。

下面是一个简单的 JSP 页面，请大家尝试阅读：

程序清单　文件 web\2\impression.jsp

```
1：<%--
2：        Document     : impression
3：        Created on : 2009-7-21, 10:14:21
4：        Author       : xuwz
5：--%>
6：
```

```
7：<%@page contentType="text/html" pageEncoding="UTF-8"
      import="java.util.*"%>
8：<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9："http://www.w3.org/TR/html4/loose.dtd">
10：
11：<html>
12：      <head>
13：            <meta http-equiv="Content-Type" content="text/html;
                     charset=UTF-8">
14：            <title>First impression</title>
15：            <link href="my.css" rel="stylesheet" type="text/css">
16：      </head>
17：      <body>
18：            <div id="wrap">
19：                  <div id="logo">
20：                        <p>JSP 印象</p>
21：                  </div>
22：                  <div id="content" >
23：                        <%
24：                        Date d1 = new Date();
25：                        out.print("今天是" + d1 + ",欢迎您!");
26：                        %>
27：                  </div>
28：                  <div id="foot">&copy;My Company  版权所有</div>
29：            </div>
30：      </body>
31：</html>
```

似乎是“新瓶装旧酒”？大家看到，这个文件中出现了 HTML 标签、CSS 代码，以及 Java 代码，这些内容对您来说可能并不陌生，陌生的是它们现在同时出现在了一个程序文件中。此外，还有一些陌生的内容如下。

- 第 1～5 行是注释代码，由符号“<%--”和“--%>”括起的内容为 JSP 的注释。
- 第 7 行是 JSP 中的页面指令，说明本网页的一些特性。
- 第 23 和 26 行的符号“<%”和“%>”，在它们之间出现的内容为采用 Java 语言编写的代码段。

另外，一个 JSP 页面中还可能出现表达式、动作等其他形式的页面元素，这些内容将在后续的学习过程中逐步接触到。

2.3　HTTP 的请求/响应模型

我们通常使用各类浏览器软件访问网站，这些浏览器或其他下载工具常常被称为“客户端”。在浏览器的地址栏中输入我们要访问的网页的地址后，位于 Internet 上的服务器会接收到请求。如果请求的是静态网页，则直接将网页内容发回给客户端；如果请求的是动态网页，则需要执行其中的动态代码段，将处理后的网页内容（类似于网页快照）以 HTML 格式发送回客户端。

客户端与 Internet 上的服务器之间是采用 HTTP 协议进行通信的，因此有必要了解一下 HTTP 协议的相关知识。

HTTP 协议即超文本传输协议（Hypertext Transfer Protocol）的简称，用于传输超文本内容。它是基于应用层的网络通信规范，即客户端与服务器进行通信时需要遵守的一种规范，广泛应用于 WWW 的全球信息服务。这里的服务器端是指提供 HTTP 服务的部分，而客户端是指浏览器或者下载工具等。

HTTP 有 1.0 和 1.1 版本，HTTP/1.1 的详细说明可以查阅 RFC2616 文档。

HTTP 协议是如何工作的呢？它采用“发起请求/发回响应”的方式进行双方的通信，即请求/响应模型（request/response），具体工作流程如下。

1．客户端请求 HTTP 服务

首先，客户端向服务器发起连接请求，服务器接受并建立通信连接。

连接建立成功后，客户端将请求信息发给服务器。一个请求主要由 4 部分组成，格式如下：

```
HTTP 请求行
请求标头
空行
消息体（可选）
```

注：HTTP 请求行和请求标头必须以<CR><LF> 作为结尾（即回车换行）。空行内必须只有<CR><LF>而无其他空格。

消息体中包含要发送给服务器的所有数据，如请求参数、Cookie 等。

2．服务端接受请求并返回 HTTP 响应

Web 服务器收到请求后，解析请求并定位指定请求资源。服务器将响应内容以特定的格式发送给客户端。一个响应由 4 部分组成，格式如下：

```
HTTP 状态行
响应标头
空行
消息体（可选）
```

消息体中包含了要显示在浏览器中的所有内容。

3．断开连接，浏览器解析响应

发送完数据后，服务端关闭连接。浏览器解析状态行、响应标头、空行，以及消息体中的响应数据等，并根据 HTML 的语法和语义对其进行格式化，在窗口中显示 HTML 格式的页面内容。

HTTP 协议是无状态的协议，即在处理完一个请求后，Web 服务器并不记住关于这个客户端或这次请求的任何信息，而只在一次请求期间建立起一个临时的通信连接，处理完成后立即断开连接。

HTTP 协议的这种“无状态性”既是优点也是缺点。一方面，由于不保存状态，使得 HTTP 累赘少，系统运行效率高，服务器应答快；另一方面，由于没有状态，协议对事务处理没有记忆能力，若后续的事务处理需要之前的处理信息，那么这些信息必须使用其他方法在协议之外进行保存。

2.4　JSP 页面的执行流程

如果客户端向服务器上的某个 JSP 资源发起了请求，此时服务器是如何处理用户的这个请求并发回响应内容的呢？

当一个 JSP 页面第一次被请求时，其处理过程如图 2-1 所示。

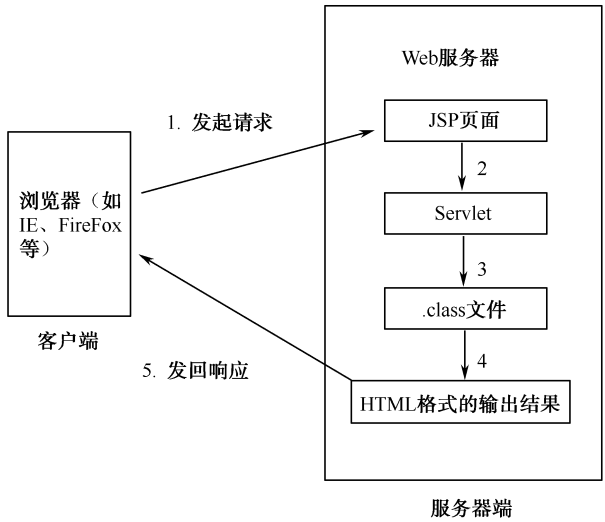


图 2-1　JSP 的执行流程

首先，被请求的 JSP 文件会被 JSP 引擎（如 Tomcat）翻译成一个 Java 源文件，这个 Java 源文件实际上是一个 Servlet 类文件；之后 JSP 引擎调用 Java 编译器把它编译成字节码文件（.class 文件），可以把这个阶段称为 JSP 页面的编译阶段；接下来是请求处理阶段，即由 JVM（Java Virtual Machine）执行此.class 文件，最后将执行的结果以 HTML 格式（即浏览器能理解的格式）返回给客户端。

若此页面之前已被请求过，且在上次编译后程序未发生修改，则不会再次编译，而是直接执行已经编译好的.class 文件。因此，JSP 页面在第一次被请求时，执行的时间相对较长。



思考与训练

一、思考题

1．静态网站与动态网站有什么区别？

2．阅读本章中介绍的第 1 个 JSP 文件，说明其中有哪些形式的 JSP 页面元素。

3．请简述 HTTP 协议的请求/响应模型。

4．从 JSP 页面的执行流程看，配置 JSP 的服务器需要安装哪些软件？

第 3 章　开发工具的安装与使用

3.1　NetBeans IDE 的安装

在安装 NetBeans IDE 之前，应先安装好 JDK（Java Development Kit）。可以分别下载 JDK 和 NetBeans 的安装包，也可以直接下载 JDK 和 NetBeans 的捆绑安装包。这里采用分别安装的方式。JDK 的下载网址是 http://java.sun.com/javase/downloads/。下载完成后，选择默认安装即可。JDK6 一般默认安装到 C:\Program Files\Java\jdk1.6.0 目录下， NetBeans IDE 或 Eclipse 安装时一般会自动搜寻这个路径。

3.1.1　下载安装包

在 NetBeans 的官方网站 www.netbeans.org 可免费下载最新版本的 NetBeans IDE。如图 3-1 所示为下载 NetBeans IDE 6.9 的网页界面。



图 3-1　下载 NetBeans IDE 6.9 的网页界面

在图 3-1 所示的网页中，清楚地列出了支持不同开发需求的安装包，可以根据自己的需要选择下载。为了开发 Java Web 项目，可选择下载 Java 或 All 安装包。这里选择下载支持开发 Java 应用的安装包。下载完成后，将得到名为 netbeans-6.9-ml-java-windows.exe 的安装文件。

3.1.2　安装 NetBeans IDE

双击得到的安装文件，启动安装程序。在如图 3-2 所示的窗口中单击“定制”按钮，可选择要安装的内容。



图 3-2　安装初始界面

单击图 3-2 窗口中的“定制”按钮后，将出现如图 3-3 所示的定制窗口。

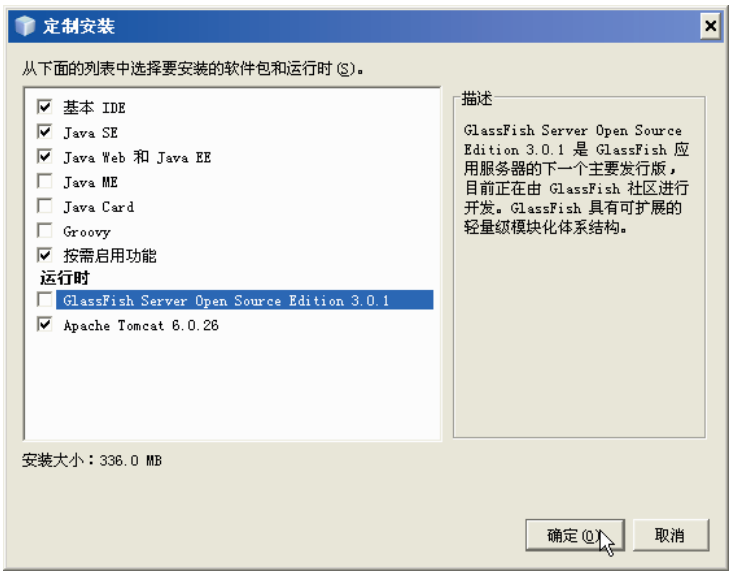


图 3-3　定制 NetBeans IDE

建议选择安装 Apache Tomcat 作为 Web 服务器，GlassFish Server 是 NetBeans 提供的另一种服务器，Apache Tomcat 和 GlassFish Server 二者至少要选择其一。选择完成后，单击“确定”按钮，将回到如图 3-2 所示的初始界面，此时在窗口中列出了所定制的安装内容，在此单击“下一步”按钮。

在出现的协议须知窗口中选择“我接受许可协议中的条款”，单击“下一步”按钮继续，将出现如图 3-4 所示的安装路径选择窗口。

第 1 个输入框内是 NetBeans IDE 的安装路径，可自行修改。需要注意的是，第 2 个输入框中应输入 JDK 的安装路径，一般来说，在已经正确安装了 JDK 的情况下，安装软件会自动

检测到其安装路径，通常不需要修改。



图 3-4　选择安装路径

接下来，根据在前面定制界面中的选择，可能还会出现设置 GlassFish Server 和 Apache Tomcat 的安装路径的窗口，请根据自己的需要进行设置。设置完毕后，将在如图 3-5 所示的窗口中显示刚刚所设置的所有路径信息。



图 3-5　安装路径信息

若路径设置无误，请单击窗口中的“安装”按钮开始具体的安装工作。此时，将出现如图 3-6 所示的安装进度显示窗口。

请等待几分钟，安装完成后，将弹出安装完成窗口，如图 3-7 所示。可以把窗口中两个选项前面的钩去掉，以节省安装时间。若选上，则会转去官方网站完成用户信息的注册，由于 NetBeans IDE 是免费使用的软件，因此可省略注册这一环节。

单击窗口中的“完成”按钮，至此，NetBeans IDE 的安装全部完成。

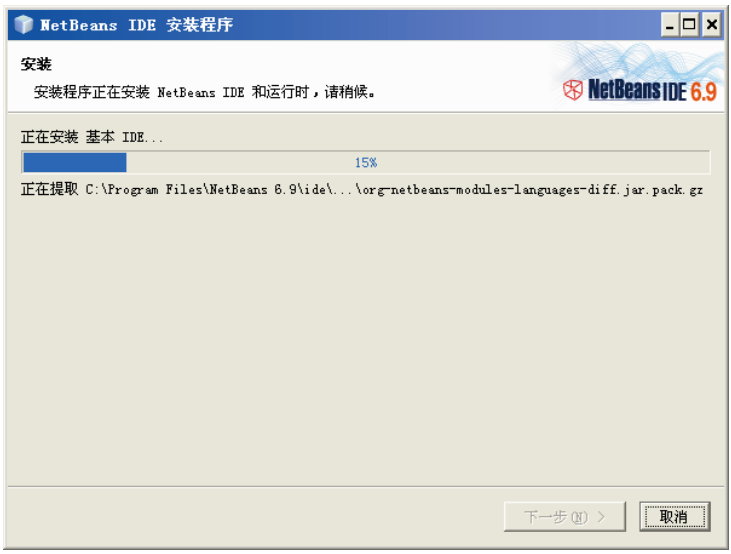


图 3-6　安装进度显示窗口



图 3-7　安装完成

3.2　创建项目与编写程序

安装完 NetBeans 后，单击桌面上的 图标，即可启动 NetBeans IDE 6.9，启动画面如图 3-8 所示。

模块装载完毕后，就进入了 NetBeans 的使用界面。首先显示的是起始页，可以将起始页上的 启动时显示 选项前的钩去掉，以后将不再显示这个页面。



图 3-8　NetBeans IDE 6.9 的启动画面

3.2.1　认识 NetBeans IDE 的界面

下面简单介绍一下 NetBeans IDE 的使用界面。为了让大家更全面地认识编写程序时最经常使用的界面，我们先创建了一个 Java Web 项目，并且运行了一个页面文件，此时的界面如图 3-9 所示，创建项目和运行页面文件的方法稍后会介绍。

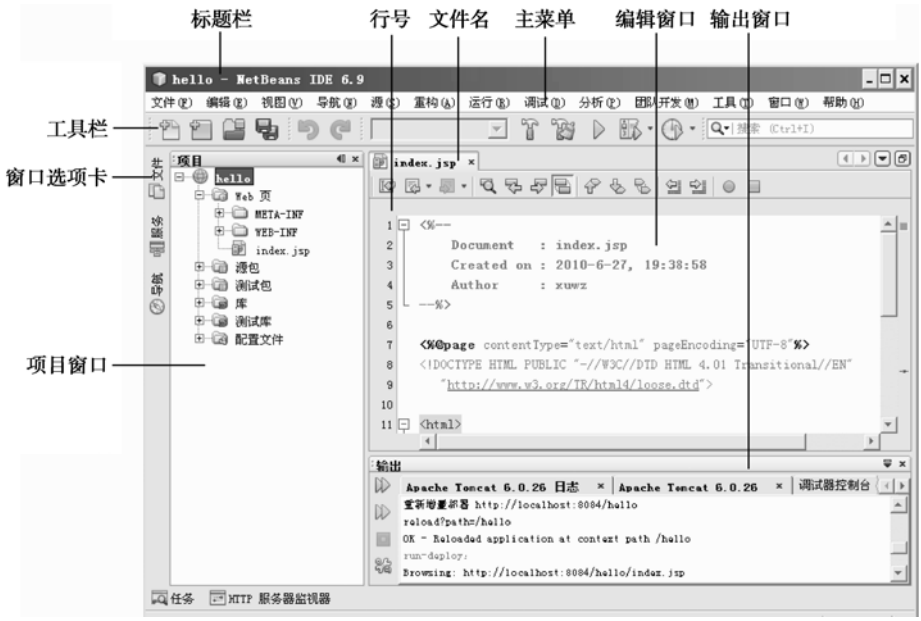


图 3-9　NetBeans 的使用界面

与一般工具软件相同，窗口的最上方是标题栏，显示了正在操作的项目的名称和开发工具的名称。标题栏的下方是主菜单，快捷工具栏紧随其下，IDE 窗口的最下方是状态行，而中部则是默认打开的几个常用窗口。用户可以通过主菜单上的“窗口”选项，自由设定要在界面上打开哪些子窗口。

左侧默认打开的窗口一般有 3 种，分别为项目窗口、文件窗口和服务窗口。在项目窗口中，分类显示项目的所有资源；在文件窗口中，显示项目中各文件的物理存储位置；服务窗口则提供对服务的管理，如 Web 服务器的管理等。

1．项目窗口

在该窗口中展开项目，可以浏览项目中的各类资源，可同时打开多个项目。其中名称以粗体字标识的项目称为“主项目”，工具栏上有专门为主项目设置的一些快捷工具按钮。

一般来说，HTML 或 JSP 等页面文件被称为“Web 页”，而后缀为.java 的类定义文件则被称为“源文件”，通常存放在源包中。需要注意的是，这里主要按资源的类别来浏览项目中的各个文件，与文件的物理存储路径是有区别的。

2．文件窗口

显示项目目录下所有文件的物理存储路径，与磁盘上的文件结构一致。

3．服务窗口

对各类服务进行管理，如数据库服务、Web 服务等，可在此启动或停止 Tomcat 服务。

4．编辑窗口

在项目窗口中选择一个文件，在文件名上双击，将弹出代码的编辑窗口。可以同时打开和编辑多个文件，通过单击编辑窗口上方的文件名称标签即可很方便地切换到另一个文件的编辑窗口中。

5．导航窗口

此窗口显示项目窗口中所选定文件的 HTML 标签结构，通过单击该窗口中的 HTML 标签，可在代码编辑窗口中快速定位到相应的代码行。

6．组件窗口

右侧一般默认打开的是组件窗口，如 HTML 的表单组件等，利用组件可以提高代码编写的效率。

7．输出窗口

界面的底部一般是输出窗口，用于显示编译、调试、运行时的各类信息。

每个窗口的右上角都有两个按钮，即最小化按钮和关闭按钮。当窗口被最小化时，会将该窗口标签收缩到原窗口的左侧、右侧或下方，当鼠标箭头移到该窗口名称上方时，窗口会自动打开，使用非常方便。图 3-10 即将项目窗口和文件窗口都最小化了，但左侧边框上仍保留了项目窗口和文件窗口的标签。

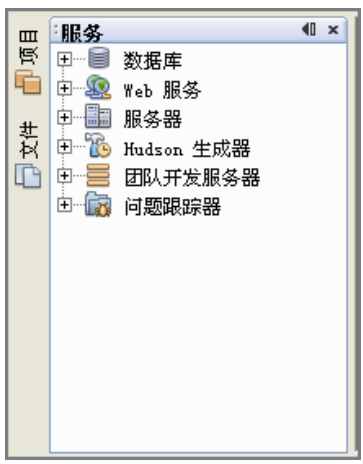


图 3-10　服务窗口

当鼠标箭头移到左侧的“项目”标签上时，项目窗口将自动展开，通过单击窗口右上角的图钉按钮，可将窗口恢复为常开的状态。

建议读者整理界面上的窗口，只保留项目、文件、服务、输出、编辑窗口，若以后还需要打开其他窗口，可以通过主菜单栏上的“窗口”菜单选择打开。

3.2.2　创建新项目

NetBeans 以项目的形式管理相关的页面及其他资源，因此首先需要创建一个新项目。下面介绍创建 Java Web 项目 hello 的过程。可以通过单击工具栏上的第 2 个按钮，或文件菜单上的“新建项目”选项来创建。系统将弹出如图 3-11 所示的项目类型选择窗口。



图 3-11　选择项目类型

我们创建的项目使用的是 JSP 技术，因此在左侧“类别”中选择“Java Web”，在右侧的“项目”中选择“Web 应用程序”，然后单击“下一步”按钮。第一次新建项目时，将激活 Java Web 功能，请稍微等待一下，之后将弹出如图 3-12 所示的窗口，在此窗口中输入“项目名称”和“项目位置”。



图 3-12　输入“项目名称”及“项目位置”

该例中将项目名称设定为“hello”。注意，当项目位置设为“D:\jsp”时，该项目的所有文件都存放于 D:\jsp\hello 下面，而不是 D:\jsp 下面，以后创建的其他项目也会将 D:\jsp 作为默认的父目录。

新创建的项目一般会自动被设为主项目，对于主项目，可以直接使用工具栏上的快捷按钮完成一些操作。如果不希望自动成为主项目，可将图 3-12 中“设置为主项目”选项前面的钩去除。若要更改主项目，可以在项目窗口中需要设为主项目的项目名称上单击鼠标右键，在弹出的右键菜单中选择“设置为主项目”即可。

单击“下一步”按钮，将弹出如图 3-13 所示的对话窗口，在此设置项目运行时所选用的服务器，这里选择“Apahche Tomcat 6.0.26”；窗口中的“上下文路径”表示访问该项目时所使用的 URL，默认与项目名称相同，保持默认值即可，最后单击“完成”按钮。



图 3-13　选择服务器

NetBeans 将为我们创建好项目的基本框架，并生成一个默认主页，文件名为 index.jsp，该文件的内容将自动在右侧的代码编辑窗口中打开。

单击菜单栏上的运行按钮，将访问主项目的默认主页（一般默认为 index.jsp），并自动打开浏览器，网页的运行结果显示在浏览器窗口中，如图 3-14 所示。在 NetBeans 中，将客户端对页面的访问称为“运行文件”。



图 3-14　默认主页的运行结果

有时，在运行页面时会弹出如图 3-15 所示的警告框，表示在运行页面时无法自动打开浏览器窗口。

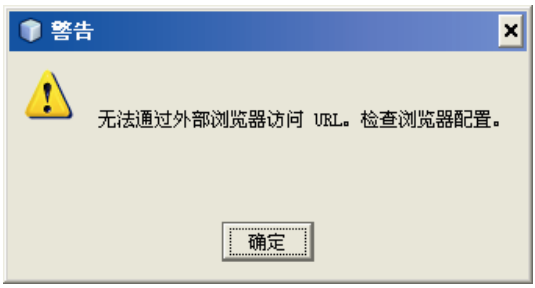


图 3-15　警告不能自动打开浏览器

此时可通过单击主菜单的“工具”→“选项”，打开 NetBeans 的系统选项设置窗口，如图 3-16 所示。在“常规”选项卡中将 Web 浏览器选项由“缺省系统浏览器”改为 IE 或其他指定的浏览器。再次运行页面，则会自动打开浏览器窗口了。



图 3-16　选择运行页面时所使用的浏览器类型

将左侧的窗口由“项目”窗口切换到“文件”窗口，可以看到如图 3-17 所示的各类文件的物理存储路径。

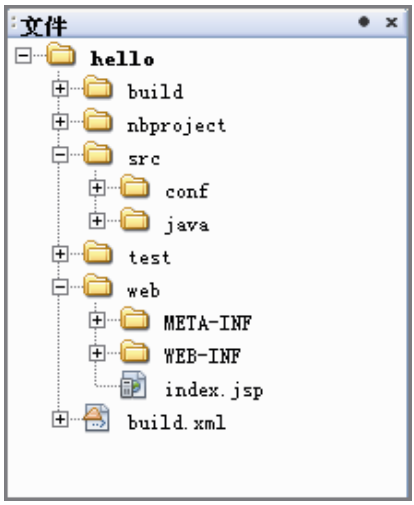


图 3-17　NetBeans 中项目的文件结构

项目 hello 中的所有文件都存储在 hello 目录下，其主要子目录的存放内容如下。

■ web 子目录：存放网页文件，如 JSP 文件、HTML 文件、JavaScript 文件、CSS 文件等，可以再创建子目录进行更细的分类；在发布网站时，该目录中的资源将全部成为网站根目录下的文件。

- src 子目录：存放源文件，项目中创建的 JavaBean、Servlet 等后缀为.java 的文件存储在 src\java 目录下；在发布网站时，src\java 目录下的所有.java 文件将被编译成同名的.class 文件，并存放在网站根目录下的 WEB-INF\classes\子目录中。
- build 子目录：存储运行时生成的结果文件，如编译生成的 Servlet 及.class 文件。
- dist 子目录：存储网站发布时需要使用的发布包。

只有使用了“生成”操作后，才会产生 build 和 dist 子目录。所谓“生成”，主要是指将项目中的 JSP 文件转换为 Servlet 类文件，将 Servlet 编译为二进制的字节码文件（后缀为.class），同时产生发布包。对于主项目，可以通过主菜单“运行”中的“生成主项目”或“清理并生成主项目”来生成项目，也可以直接单击工具栏上的快捷按钮和；若非主项目，则可在项目名称（先切换到项目窗口中）上单击鼠标右键，再在弹出的右键菜单中选择相应的操作项进行处理。

此时在 NetBeans 中新建的 JSP 页面，其代码均与项目 hello 中的 index.jsp 页面相同，因为所有的 JSP 文件都是按照工具中自带的 JSP 模板创建的，可以对该模板进行修改定制。

3.2.3　修改默认模板

在 NetBeans 中编写的各类程序都有自己的默认模板，其默认的 JSP 模板中包含有 <h1>Hello World!</h1> 这句代码，因此在每个新生成的 JSP 文件中无一例外地都会出现它，实际上一般的页面并不需要这句代码。我们可以通过修改 JSP 的模板一次性地将此去除。

在主菜单的“工具”组中选择“模板”，打开如图 3-18 所示的“模板管理器”窗口。



图 3-18　“模块管理器”窗口

展开“Web”节点，单击其下的“JSP”选项，单击下方的“在编辑器中打开”按钮，可能会出现如图 3-19 所示的“警告”窗口，单击“确定”按钮，于是模板内容将显示在编辑窗口中。

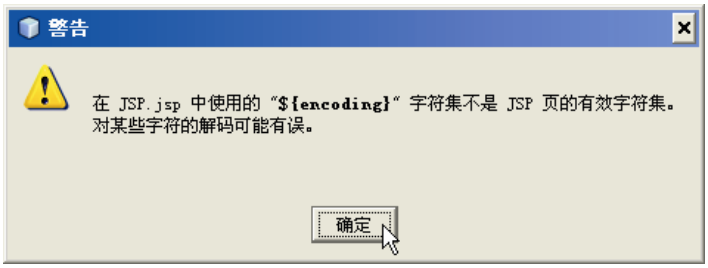


图 3-19　打开模板时关于字符集的警告

NetBeans 的模板是采用 FreeMarker 编写的，可以对其进行修改。其中以${}形式出现的表达式表示模板中使用的变量，常用的几种变量格式如下所示。

- ${date} 按系统默认的格式插入当前日期。
- ${time} 按系统默认的格式插入当前时间。
- ${encoding} 插入默认的编码字符集。
- ${name} 插入正在编辑的文件的名称。
- ${nameAndExt} 插入正在编辑的文件的全名（含后缀）。
- ${package} 插入文件所在的包名。
- ${user} 插入系统用户名。

有兴趣的读者可参考 FreeMarker 的相关资料。

保存所做的修改后，以后新建的 JSP 页面将自动采用此模板生成。

3.2.4　创建 JSP 页面

下面在 hello 项目中添加一个 JSP 文件，看看新生成的 JSP 页面文件是否采用了新模板。

1．新建 JSP 文件

先在项目窗口中选择项目“hello”，单击项目名下方的“Web 页”，以预先确定新建文件的存储位置，然后单击工具栏上最左边的第 1 个按钮，在出现的类型选择窗口（如图 3-20 所示）中选择“Web”和“JSP”，单击“下一步”按钮。

在弹出的对话窗口中输入文件名，注意不要填写后缀名，单击“完成”按钮。

新创建的文件将自动在代码编辑窗口中打开，可以看到，程序采用了新的模板风格。

2．添加 Java 代码段

下面在页面文件中加入一些 Java 代码，必须用符号“<%”和“%>”括起来。在文件 newjsp.jsp 的编辑窗口中的标签<body>和</body>之间加入如下的代码：

```
<%
int i,j;
      for   i=0;i<3;i++)
      {for (j=0;j<2*i+1;j++)          out.print("*");
        out.print("<br/>");
      }
%>
```

3．语法错误标识

从图 3-21 可以看到，若输入的代码有语法错误，则该行代码的下方会被加上红色的波浪

线，并在左侧的灰色边栏中出现警告标志。修改为正确的代码后，这些警告标识将自动消失。



图 3-20　选择文件类型

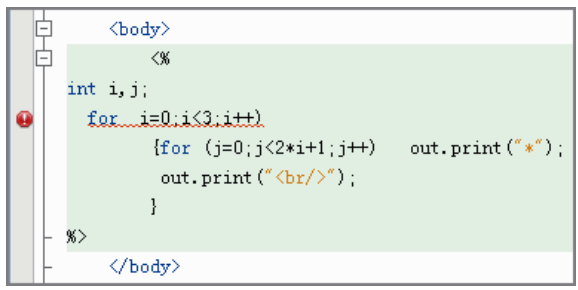


图 3-21　自动提示语法错误

4．自动格式化代码

当代码排列不整齐时，可以在代码编辑窗口单击鼠标右键，在右键菜单中选择“格式化代码”，代码将自动按缩进模式排列整齐。

5．显示行号

为了方便调试修改代码，应该将各行代码的行编号显示出来。若行号未显示，可右击（用鼠标右键单击）代码编辑窗口左侧的灰色边栏，在右键菜单中选择“显示行号”，即可将行号显示出来，如图 3-22 所示。

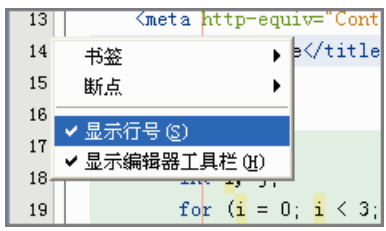


图 3-22　显示行号的设置

3.2.5　运行 JSP 页面

前面介绍了可以通过工具栏上的快捷按钮去运行一个主项目的默认主页，那么如何运行一个普通项目中的普通页面文件呢？这时可以采用下面两种比较快捷的方式。

- 在页面文件的代码窗口中单击鼠标右键，在右键菜单中选择“运行文件”，或者将光标先定位在该文件的代码编辑窗口中，然后按快捷键“Shift+F6”。
- 在项目窗口中找到该文件，在文件名上单击鼠标右键，在弹出的右键菜单中选择“运行文件”。

在编译的过程中，将在输出窗口中显示相关的编译信息，在状态行也可以看到处理任务时的状态信息，如图 3-23 所示。



图 3-23　页面成功生成时的控制台信息

该页面运行的结果如图 3-24 所示。



图 3-24　页面的运行结果

从图 3-24 中的地址栏可以看出，跟随 NetBeans IDE 捆绑安装的 Apache Tomcat 的运行端口是 8084。地址栏中的“localhost”表示服务器位于本机上，“8084”表示服务器的运行端口号，“/hello”是项目的访问路径，默认情况下与项目名称相同，“newjsp.jsp”则是被访问的页面文件的名称。

3.2.6　认识 JSP 的页面元素

除了 HTML 标签和 Java 代码外，JSP 页面中还能出现哪些形式的元素呢？下面是一个比较典型的 JSP 页面的代码：

```
1：<!-- 一个典型的 JSP 页面-->
2：<%-- 文件功能：验证登录--%>
3：<%@ page contentType="text/html;charset=utf-8" %>
4：<html>
5：<head>
6：<title>验证中...</title>
```

```
7：</head>
8：<body>
9：<%@ include file="head.jsp"%>
10：<jsp:useBean class="com.sunny.Users" id="u1" />
11：<jsp:setProperty name="u1" property="*" />
12：<%
13：    if (u1.check())
14：        out.print("hello,"+u1.getUname());
15：    else  {
16：        out.print("<script>alert('输入有误，请重新输入.'); </script>");
17：%>
18：<jsp:forward page="login.jsp" />
19：<% } %>
20：<%@include file="foot.jsp"%>
21：</body>
22：</html>
```

在上面的例子中，可以看到除了 HTML 代码外，还有许多不同的符号，它们分别表示 JSP 中的不同语法元素，简单说明如下。

- 第 1 行为 HTML 格式的注释。
- 第 2 行为 JSP 格式的注释。
- 第 3 行、第 9 行分别为 JSP 的 page 指令和 include 指令。
- 第 10 行、11 行、18 行为 JSP 的动作。
- 第 12～17 行为 JSP 的代码段。

一个 JSP 页面中主要包含 JSP 的指令、动作、声明和代码段等元素，详细信息可参考第 17 章的内容。

3.2.7　添加自定义的模板

除了修改 IDE 自带的默认模板外，我们还可以根据需要添加自定义的模板。最简便的方式是先复制一个已存在的模板，然后修改其内容。

1．添加带表单的 JSP 页面模板

由于在开发中经常用到输入表单，因此可以制作一个包含表单的页面模板。

在如图 3-25 所示的“模板管理器”窗口中先选中“Web”→“JSP”，然后单击“复制”按钮，将在 Web 类别下得到一个名称为“JSP_1.jsp”的模板，将其重命名为“JSP_Form.jsp”（输入新的名称后需按回车才能生效）。然后打开此模板文件，在其<body>和</body>之间加入如下所示的表单内容：

```
<body>
    <form method="POST" action="">
        <input type="text" name="" value="" /><br/>
        <input type="submit" value="确定" />
        <input type="reset" value="重置" />
    </form>
</body>
```



图 3-25　复制模板

最后，保存修改内容即可。依照上述的方法，大家还可根据需要创建自己的更多模板。

2．使用新添加的模板

单击工具栏最左边的第 1 个按钮，新建一个文件，在弹出的文件类型选择窗口中，将看到以新模板名称命名的文件类型，如图 3-26 所示。

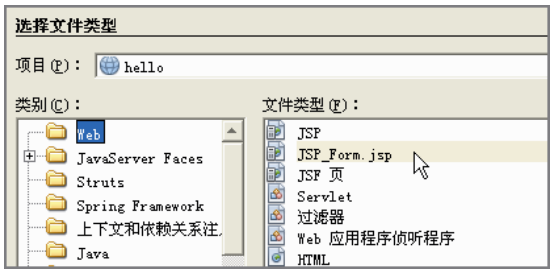


图 3-26　新模板

选择新建一个此类型的页面 form1.jsp，如图 3-27 所示，可以看到新建的文件中已经自动包含了表单内容。正确地使用模板可以提高代码编写的效率。

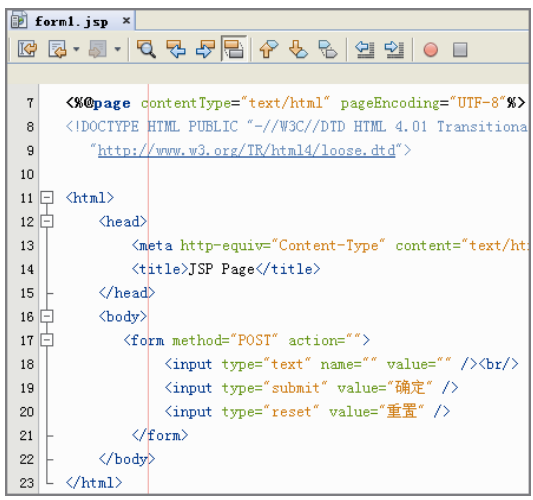


图 3-27　根据表单模板创建的 JSP 页面

3.2.8　打开/关闭项目

在项目窗口中可以通过右击项目名称，在右键菜单中选择“关闭”来关闭该项目，这时项目窗口中将不再显示该项目。

若需要再次打开，则可以在主菜单项“文件”中选择“打开项目”，或者单击工具栏上的按钮，在磁盘上找到项目的存放位置，打开即可。若希望打开的项目成为主项目，应将“作为主项目打开”选项勾选上，如图 3-28 所示。



图 3-28　打开已存在的项目

对于一个已经打开的项目或文件，只需将鼠标箭头停留在其名称上，即可显示出它在磁盘上的存放位置，如图 3-29 所示。

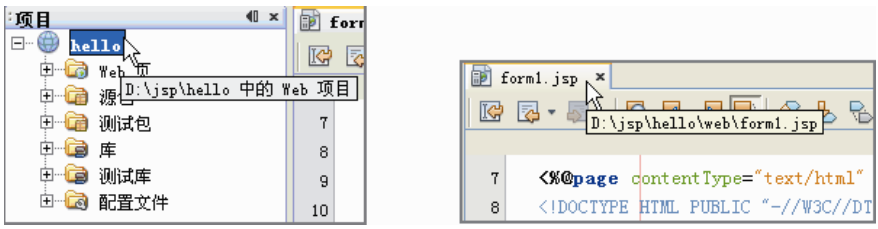


图 3-29　查看项目或文件的存放路径

3.3　在 NetBeans IDE 中调试程序

编写程序时，错误是难免的，高效地排查这些错误，是程序员必备的基本技能。下面介绍对于不同的程序错误采取的一般性排查方法。

1．语法错误

程序的语法错误往往是比较容易查找和修改的。当程序中存在语法错误时，NetBeans 通常都会在编辑窗口直接提示，如在出错的代码下方以红色的波浪线标识。当我们将光标移到有错的代码行上时，一般会出现更为详细的提示信息，可根据这些提示信息改正代码。

2．运行时异常

若程序不存在语法错误，但在运行时出现了异常，可查看控制台上的输出提示信息，以期找到产生错误的原因。

如图 3-30 所示，可在 NetBeans 的输出窗口中单击“Apache Tomcat 6.0.26 日志”选项卡，查看在控制台上输出的异常信息。若在此处难以查找到错误的原因，则应采用下文介绍的断点跟踪调试法进行查找。

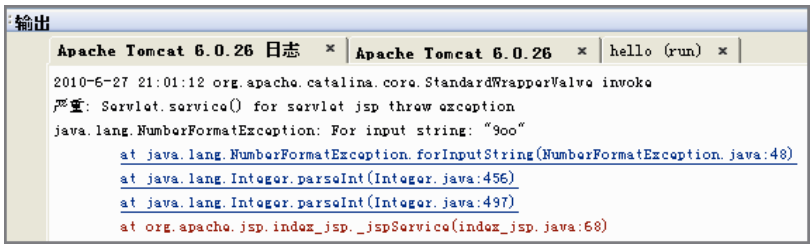


图 3-30　控制台输出的异常信息

3．逻辑错误

当程序运行的结果与所预期的不同，即产生了逻辑上的错误时，往往较难查找原因。这时最有效的手段是在程序中设置断点，然后跟踪程序的运行过程，通过观察运行过程中变量的值或表达式的值是否与预期的相同，或者每条代码是否正常执行，来判断引起错误的代码。一般步骤如下。

1）设置断点

所谓断点，是指程序中的一行代码，当程序被调试执行时，遇到该行代码则会暂停。一个程序中可以设置多个断点。

要使一行代码成为断点，只需在相应的行号处单击鼠标左键，即可以看到行号处出现了红色的小方块，且代码行也以红色背景条醒目地标识，如图 3-31 所示。

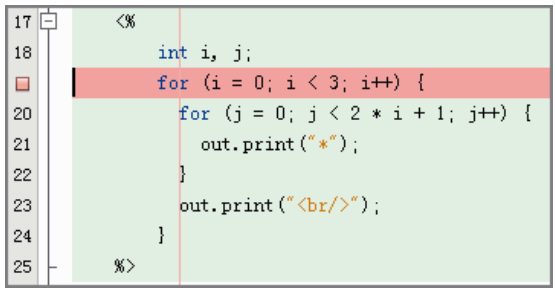


图 3-31　程序中的断点

注意：并不是任意的代码行都能成为有效断点，断点行代码必须是可执行的。如图 3-31 中的第 18 行，只是变量定义语句，无实际的执行内容，因此若把它设为断点，则是无效的断点，调试时也不会在此暂停。

2）启动调试，跟踪程序的执行

设好断点后，就可以跟踪一个程序的执行过程了。

可以有以下两种方法启动调试模式。

■ 将光标先定位在被调试文件的代码编辑窗口内，而后在 NetBeans 的主菜单上选择“调试”→“调试文件”。

■ 在项目窗口中选中要调试的文件，右击文件名，在右键菜单中选择“调试文件”。

不管使用哪种方法，都将出现如图 3-32 所示的界面。

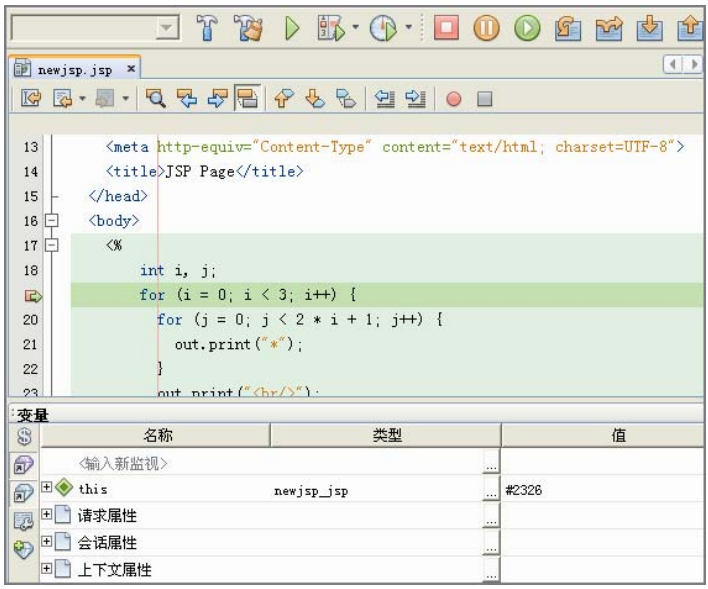


图 3-32　跟踪调试界面

可以看到，原来设为断点的那行代码被以绿色背景条标识，且左侧有一个绿色的小箭头，表示该行代码即将被执行，下文简称其为“当前行”。

处于调试状态时，工具栏上将出现一组调试专用的按钮，往往也可使用快捷键完成这些按钮提供的功能。最常用的快捷功能键有“F8”、“F7”和“F4”键。

按功能键“F8”键，或工具栏上的按钮，将执行当前行的代码，并且在下一行代码处停下。

若一行代码中进行了方法的调用，且希望对方法内部的代码进行跟踪，则可以在执行该行代码时按功能键“F7”键。

按下功能键“F4”键，将执行从当前行到光标所在处的代码段，系统会在光标所在行停住。

在本例中，可以持续使用“F8”键，代码将逐行被执行。

3）查看变量值

在编辑框的下方，是变量窗口，在此会实时显示调试时程序中所有变量的值。

在逐条执行代码的同时，可以看到某些变量的值也在同步变化，如图 3-33 所示。在调试前面的 newjsp.jsp 页面的代码时，可在变量窗口中观察到其中的变量 i 和 j 的值。

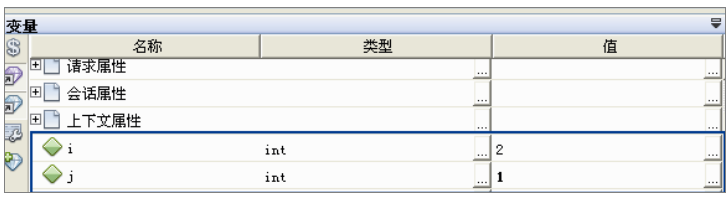


图 3-33　变量窗口

另一个快速观察变量值的方法是，在代码窗口中将鼠标箭头移到变量名的上方稍作停留，也会显示该变量的当前值。图 3-34 显示的是鼠标光标移到变量名 i 上方的情形，提示框中显示目前 i 的值是 2。

通过观察变量的值，并与预期的正确值进行比较，往往可以比较容易地判断出出错的代码。此外，还可通过监视窗口来观察一些表达式的值。

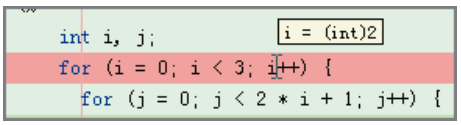


图 3-34　快速显示变量值

4）结束调试

若希望回到编辑状态，或者想重新运行程序，则应先退出调试状态。按工具栏上的 停止按钮即可结束调试。



思考与训练

一、思考题

1．编写程序时遇到的错误通常有哪几种？可采取哪些方法解决？

2．一个 Java Web 项目的文件组织结构是怎样的？项目中各文件夹分别用来存储哪些文件？请对一个项目执行“生成项目”操作，并观察 build 目录、dist 目录中的文件内容。

二、综合实训

1．新建一个项目，并为该项目添加一个 JSP 文件，以输出您自己的姓名和当天的日期，尝试运行该文件。

2．有一个 JSP 页面文件，其主要代码如下：

```
<body>
    <%    int a[]={1,2,3,4};
          int b[4];
          for (int i=0;i<4;i++){
                b[i]=a[4-i];
          }
          for (int i=0;i<4;i++)
                out.println(b[i]+"<br/>");
    %>
</body>
```

请在 NetBeans 中设置断点进行跟踪调试，找出错误的原因并改正。

第二部分　在开发中学习

该部分是本书的核心内容，描述了论坛网站前台功能模块的开发全过程。在逐个完成论坛的注册、登录、浏览帖子、发表回复等功能的过程中，由浅入深地讲解了 JSP 技术的相关知识，如 JSP 的指令、动作、内建对象、JDBC 访问数据库的方法等。其中，第 10 章以注册功能的改进为例，介绍了 JavaBean、Servlet、文件上传等较为高级的内容。具体章节如下。

第 4 章　论坛网站的设计
第 5 章　制作静态网页
第 6 章　注册功能的实现
第 7 章　登录功能的实现
第 8 章　帖子浏览功能的实现
第 9 章　回帖功能的实现
第 10 章　功能的增强与改进
第 11 章　服务器的配置与网站的发布

第 4 章　论坛网站的设计

在第 1 章中已经对论坛的功能需求进行了分析，通过前面的分析得知，普通用户应具有注册、登录、浏览帖子、发帖、回帖、修改帖子，以及修改个人资料的功能，如图 4-1 所示的 UML 活动图描绘了普通用户在论坛中的基本活动。

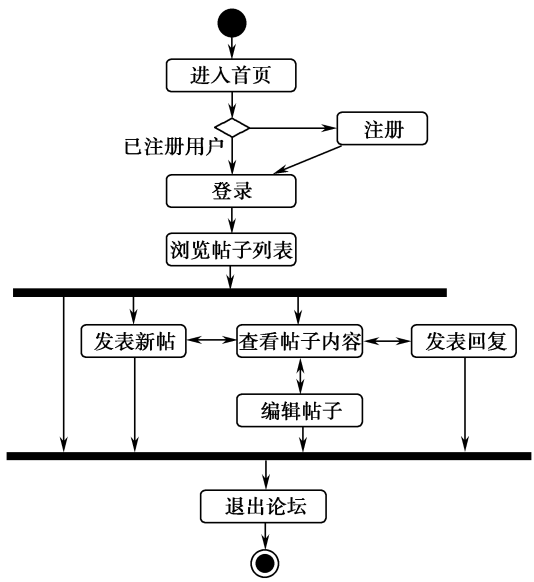


图 4-1　前台模块的活动图

4.1　功能模块设计

提供给普通用户使用的前台功能模块设计如下。

1．用户注册

论坛要求所有用户均为注册用户。未注册的新用户可在注册页面 reg.jsp 中填写自己的用户名、密码、E-mail、头像等个人信息。用户名在整个论坛中应该是唯一的，因此当用户完成信息的填写并提交给处理页面 chkReg.jsp 后，应在 chkReg.jsp 中判断此用户名是否已被他人注册过了，若未被使用，则将用户信息保存到数据库中，并给出相应的提示信息。

2．用户登录

用户可使用注册时填写的用户名和密码登录论坛，论坛的所有功能均需登录才能使用。当用户在登录页面 login.jsp 中输入用户名和密码，并且提交信息后，处理页面 chkLogin.jsp 根据用户输入的信息查询数据库，完成用户信息的检测工作。用户登录成功后，将自动跳转到论坛的首页。

3．帖子列表显示

论坛首页 index.jsp 以列表的形式显示论坛主题帖的标题、作者、点击数等概要信息，当用户单击每一个主题帖的标题超链接后，将跳转到帖子详情页面 postDetail.jsp，在这里可查看该帖子的内容及回复帖信息。此时，该主题帖的点击数应加 1。

4．帖子详情浏览

在帖子列表页面单击主题帖的标题后，可在页面 postDetail.jsp 查看主题帖的详细信息，包括主题帖的作者（用户名、级别、注册时间等信息）、标题、内容、发布时间，以及本主题帖所有的回复帖等信息。

5．发帖

登录用户可以发表新的话题，在发表新帖页面 newPost.jsp 中，用户应填写帖子的标题和内容。待信息提交给处理页面 savePost.jsp 后，该主题帖的标题、内容、作者及发帖时间等信息将保存到数据库中。对于新发表的帖子，其点击数和回复数均默认为 0。

6．回帖

登录用户在浏览帖子的同时，可以针对该主题帖发表自己的回复内容。用户通过回帖页面 newReply.jsp 输入回复帖的标题和内容。提交后，由处理页面 saveReply.jsp 将回复帖的信息保存到数据库中，包括回复帖的标题、内容、作者、发表时间，以及相关主题帖的 ID。此外，相关主题帖的回复数应加 1。

7．编辑帖子

登录用户在浏览帖子时，若某个主题帖或回复帖的作者为当前用户，则应提供修改帖子的超链接。单击此链接，可打开帖子修改页面 editPost.jsp，用户可以在此修改自己发表的主题帖或回复帖，包括标题和内容。修改完成后，提交到 saveEditPost.jsp 页面，以更新数据库中的帖子信息。

8．修改个人信息

登录用户可通过个人资料更新页面 editInfo.jsp 修改自己的密码、头像等信息。输入新的内容后，提交到 saveEditInfo.jsp 页面，完成个人资料的更新。

9．退出系统

退出网站，即注销登录用户的信息，此后需要重新登录后才能再次访问论坛。退出处理页面为 exit.jsp。

4.2　页 面 列 表

综上所述，为了给论坛的普通用户提供基本的操作功能，应编写如表 4-1 所示的 JSP 页面。

表 4-1　网站前台模块页面列表

编　　号	功 能 模 块	页 面 名 称	说　　明
1	帖子列表显示	index.jsp	为登录用户显示帖子标题等概况信息列表
2	登录	login.jsp 及 chkLogin.jsp	login.jsp：登录信息填写页面 chkLogin.jsp：登录验证处理页面
3	注册	reg.jsp 及 chkReg.jsp	reg.jsp：注册信息填写页面 chkReg.jsp：注册处理页面
4	个人信息修改	editInfo.jsp 及 saveEditInfo.jsp	editInfo.jsp：修改个人信息的页面，可修改头像或用户密码等 saveEditInfo.jsp：更新数据库中的个人资料
5	发帖	newPost.jsp 及 savePost.jsp	newPost.jsp：发表新帖的输入界面 savePost.jsp：保存帖子
6	回帖	newReply.jsp 及 saveReply.jsp	newReply.jsp：发表对某个帖子的回复帖 saveReply.jsp：保存回复帖
7	浏览帖子	postDetail.jsp	显示帖子的详细内容，含主题帖及回复帖的信息
8	修改帖子	editPost.jsp 及 saveEditPost.jsp	editPost.jsp：帖子信息修改页面 saveEditPost.jsp：保存修改信息
9	退出	exit.jsp	退出系统，将回到登录界面

上表仅列出了需要编写的主要页面，一些辅助页面将在学习的过程中逐步添加，这里不一一列出。

4.3　数据库结构设计

在论坛网站的前台模块中，若帖子不分版块，则相关的实体主要有用户和帖子，而帖子又可分为主题帖和回复帖。因此一个简单的单版块论坛需要保存的主要信息有用户信息、主题帖信息、回复帖信息。可以相应地创建 3 张表来保存上述信息。当然，也可将主题帖与回复帖合并为一张表，但合并后在一定程度上会增加编程的复杂度。

这里先采用 Access 数据库创建所需的表，数据库名称为 bbs，各表的结构如下。

1．用户信息表 users

该表除了保存用户的用户名和密码外，还应记录用户的注册时间、头像图片的文件路径、

E-mail，以及用户级别等信息。若需要计算用户的积分，则还应添加积分列。该表中的注册日期也可设为日期型，如图 4-2 所示。

users : 表

字段名称	数据类型	
uname	文本	用户名，主键，最大长度为20个字符
upw	文本	用户密码，最大长度为12个字符
usex	文本	性别，2个字符，默认值为"密"
udate	文本	注册时间
uemail	文本	e-mail地址，长度不超过50个字符
ulevel	数字	用户的级别，默认值为1，表示为普通用户
ulogo	文本	用户的头像文件路径，长度不超过100个字符

图 4-2　用户信息表的结构

2．主题帖信息表 posts

主题帖信息表是用来保存用户发表的新话题帖的，除了保存标题、内容、作者和发表时间这些基本信息外，为了统计帖子的热门程度，还应记录帖子的点击数和回复帖的数量。此外，为了简化程序的处理，也可将最后对此帖发表回复的作者和发表时间也一并记录下来。因此得到如图 4-3 所示的表结构。

posts : 表

字段名称	数据类型	
pid	自动编号	主键，主题帖的ID号，由自动自动递增生成
ptitle	文本	帖子的标题
pcontent	文本	帖子的内容
pauthor	文本	作者
pdate	文本	发表时间，若曾被编辑，则为最后的编辑时间
preplies	数字	回复数
pclicks	数字	点击数
plastUpdate	文本	最后回复本帖的时间，若无回复，则为发帖的时间
plastAuthor	文本	最后回复本帖的作者，若无回复，则为楼主

图 4-3　主题帖信息表的结构

发帖的作者应为已登录的用户，因此列 pauthor 是与用户信息表 users 关联的外键。在 Access 数据库中，文本字段的最大长度是 255，故帖子内容的最大长度是 255，若改用其他数据库，则可相应地增加此长度。

若要添加版块信息，也可在该表中添加一列，表示所属版块的版块号或版块名称。

3．回复帖信息表 reply

因为回复帖总是与某个主题相关的，因此在记录回复帖的信息时，应同时记录与该回复帖相关的主题帖的 ID，由表中的 pid 列表示。发表回复的时间（列 rdate）也可设为时间类型，如 Access 的 timestamp 类型，则默认取值可设为当前的系统时间，如图 4-4 所示。

reply : 表

字段名称	数据类型	
rId	自动编号	回复帖的id，由系统自动生成，主键
pid	数字	相关主题帖的id，外键，参照主题帖表posts中的主键
rTitle	文本	回复帖的标题
rcontent	文本	回复帖的内容
rauthor	文本	作者，应为users表中已注册的用户
rdate	文本	发表回复的时间

图 4-4　回复帖信息表的结构



思考与训练

一、思考题

1．若需要为论坛分设不同的版块，且每一版块设立 1～2 个版主，此时数据库结构应如何设计？

2．若要为普通会员设立等级，当会员登录、发帖、回帖时，可分别得到不同数目的积分，并根据其积分设置会员的等级，则数据库结构应如何设计？

3．若需要为会员添加一对一的短消息交流功能，则相应的表结构应如何设计？

第 5 章　制作静态网页

5.1　制作网站的原型

本论坛的页面不多，可为所有页面采用相同的布局，即页面上方显示网站 LOGO 及导航条，中间为内容显示区域，页脚显示版权提示、友情链接等信息。布局如图 5-1 所示。



图 5-1　页面整体布局原型

为了对各页面的布局及操作流程有一个清晰的了解，可以先制作各网页的原型。首页原型显示帖子的主题列表，如图 5-2 所示。



图 5-2　首页帖子列表显示的页面原型

单击该页面上帖子的标题后，可查看该帖的详细情况，帖子详情页面的原型如图 5-3 所示。



图 5-3　帖子详情页面原型

帖子列表页面和详情页面上都提供了发表新帖的相关按钮，单击该按钮后，将打开发表新帖子的页面，该页面原型如图 5-4 所示。



图 5-4　发表新帖页面原型

发表回复帖的页面与此相似，这里不再一一列出其他页面的原型。不过，在动手制作网页前，绘制每个页面的原型可以帮助自己整理思路，同时对网站的整体处理流程有一个更为清晰的认识。此外，直接将网页原型呈现给客户，也可使开发方与客户之间的沟通更为顺畅，使需求更加明确。

5.2　制作注册页面

为了收集用户的基本信息，需要为新会员提供一个注册页面，我们需要完成以下任务。

- 使用 HTML＋CSS 制作输入页面。
- 使用 JavaScript 完成输入数据的前端验证。

下面介绍 HTML、CSS，以及 JavaScript 的基础知识。

5.2.1　HTML 的常用标签

当我们打开一个网页时，常常会被其令人眩目的效果所吸引，而实际上，构成网页的元素大多是文字、链接、图片和 Flash 等，这些元素是通过 HTML 的标签来描述的。

这里只介绍几个最主要的标签，其他标签可参考本书第 4 部分的相关内容，或静态网页设计的相关书籍。

1．表格系列标签

该系列的标签可在页面上绘制由行和列组成的表格，并在单元格中显示文字、超链接、图片等其他元素。

- <table></table>标签对表示一个表格。
- <tr></tr>标签对表示表格中的一行。
- <td></td>标签对表示行中的一个单元格，标签对之间的内容显示在该单元格中。
- <th></th>标签对表示标题行中的一个单元格，一般为居中加粗显示。
- <caption></caption>标签对表示显示在表格上方或下方的表格标题。

【例 5.1】　表格标签的使用一。

该例演示了表格基本标签的使用方法，显示结果为 CD 的库存信息表，代码如下：

程序清单　文件 web\5\html\table1.html

```
1：<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2：<html>
3：      <head>
4：           <title>表格示例一</title>
5：           <meta http-equiv="Content-Type" content="text/html;
               charset=UTF-8">
6：      </head>
7：      <body>
8：           <table  border="1"  cellspacing="0"  cellpadding="5">
9：                <caption>CD 库存表</caption>
10：                 <tr><th>歌星</th><th>库存量</th><th>单价</th></tr>
11：                 <tr><td>张学友</td><td>1200</td><td>120.00 元</td></tr>
12：                 <tr><td>李克勤</td><td>680</td><td>80.00 元</td></tr>
13：           </table>
14：      </body>
15：</html>
```

CD库存表

歌星	库存量	单价
张学友	1200	120.00元
李克勤	680	80.00元

图 5-5　表格的使用

<table>标签中的属性“border="1"”表示表格边框的粗细为 1px，cellspacing 属性表示单元格之间的空隙，cellpadding 属性表示单元格中的内容与单元格边框的留白距离。程序运行效果如图 5-5 所示。

还可通过<td>标签的 colspan 或 rowspan 属性设定该单元格所跨越的列数或行数，以达到合并单元格的效果，请看下面的例子。

【例 5.2】　表格标签的使用二。

该例演示了表格的跨列属性 colspan、水平对齐属性 align、宽度

属性 width 的使用方法，主要代码如下：

主要代码　文件 web\5\html\table2.html

```
1：    <body>
2：        <table border="1" width="300">
3：            <tr align="center">
4：                <td colspan="2">星期一</td>
5：            </tr>
6：            <tr>
7：                <td align="center">上午</td>
8：                <td>下午</td>
9：            </tr>
10：            <tr>
11：                <td>语文</td>
12：                <td>数学</td>
13：            </tr>
14：        </table>
15：    </body>
```

案例效果如图 5-6 所示。

星期一	
上午	下午
语文	数学

图 5-6　合并单元格

可以看出，第一行的“星期一”所在的单元格占据了 2 列的位置，也就是合并了左右 2 个单元格。该例还设定了<tr>标签的水平对齐属性 align，即设定某一行数据的水平对齐方式，可取值为 center、left 或 right，分别表示居中、靠左对齐、靠右对齐，默认值是 left。<td>标签的 align 属性则仅表示该单元格的水平对齐方式，也可通过 CSS 来设置。

表格中还可再嵌套表格，即某个单元格的内容是一张表格，通过表格的嵌套可以实现页面的布局。当然，页面的布局目前更多是通过 DIV+CSS 的方式来实现的，表格则一般被应用于数据的显示。

在实际开发中，可借助于网页制作工具软件 Dreamweaver 来实现表格效果，但应对上述标签有一定的认识。

2．表单系列标签

在动态网站的开发中，表单系列的标签是最常用的，这些标签在页面上通常呈现为各种形态的输入元素，由表单标签<form>...</form>将它们组合成一张表，用于收集用户填写的信息。

1）form 标签

格式：

```
<form    action=""    method="" name=""> </form>
```

form 标签对表示一个表单的开始和结束，属性说明如下。

■ action 属性：表示接收本表单数据的资源名称（通常是一个 JSP 页面，写出该页面的相

对路径即可），就好像递交驾驶证年审表一样，该属性表示所填写的表单数据要交给哪个“窗口”处理。当 action 值为空或未指出时，表示提交给当前页面处理。

■ method 属性：表示提交数据的方式，可取“post”或“get”值，默认值是“get”。取“get”方式时，提交的数据将附在网页地址（即 URL）后提交给接收页面。显然，这种传送方式的安全性较差。此外，由于 URL 的长度是有限制的，所以以这种方式传送的数据其长度也必然受到限制。一般建议取“post”值。

■ name 属性：表示该表单的名称，若不需要可以省略。

注意：

在英语中，“form”是“表”的意思，切勿把“form”写成“from”。

在标签<form>和</form>之间，可以包含多个输入元素，它们一般由<input>标签、<select>标签或<textarea>标签表示。

2）input 标签

格式：

```
<input  type=""  name=""  id=""  value="" >
```

input 标签是最常用的输入标签，其属性说明如下。

■ name 属性表示该输入元素的名字，即向服务器提交的请求参数的名称，通常应起一个有意义的名字。

■ id 属性表示该元素在页面中的唯一标识名，CSS 可以通过 id 来控制该元素的样式，而 JavaScript 代码也是通过 id 来查找页面上的某个元素的。一般来说，在不引起混淆的情况下，可以将 name 和 id 设为相同的值。

■ value 属性表示该输入元素的初始值，默认值为空字符串。

■ input 标签的 type 属性可取不同的值，在页面上表现为不同的形态，如表 5-1 所示。

表 5-1　input 标签的 type 属性

type 属性的取值	类 型 说 明
text	单行文本框
password	密码输入框
radio	单选按钮
checkbox	复选框
file	文件选择框
hidden	隐藏域
submit	提交按钮
reset	重置按钮
button	普通按钮
image	图片按钮

3）type 属性补充说明

■ type 取“hidden”值时，表示该 input 元素在页面上不显示，常用于隐蔽地提交一个已知的数据。

- 当 type 取“radio”值时，可为用户提供多选一的输入，这一组选项的 name 属性必须设为相同值，即名字应相同。
- 使用“radio”或“checkbox”为用户提供选择项时，应设定该 input 标签的 value 值，否则，当用户选中某个选项时，所取的值默认为“on”。
- 当 type 为“submit”（提交按钮）或“reset”（重置按钮）时，value 属性表示显示在按钮上的文字。这两个按钮有特殊的作用，即当用户单击提交按钮时，会将表单中的数据提交给 form 标签的 action 属性所指定的接收页面去处理，而当用户单击重置按钮时，各输入框的值将重新恢复为初始值。通常一个表单中应有一个提交按钮。

【例 5.3】　input 标签的 type 属性测试。

新建一个页面，在其标签<body>和</body>之间输入下面的代码：

```
1：<form>
2：    <input type="text" name="abc" />
3：</form>
```

运行此页面，将得到一个单行的文本输入框。请按照表 5-1 中 type 属性的各种取值，依次修改此例中 input 标签的 type 属性值，观察页面的显示结果。

4）select 标签

格式：

```
<select name="名称">
    <option selected value="取值 1">选项 1</option>
    <option>选项 2</option>
    ......
</select>
```

select 标签在页面中呈现为下拉框或列表框的形式，通常与<option>标签结合，每一对<option></option>标签表示框中的一个选项，标签<option>和</option>之间的内容将显示在下拉框或列表框中。<option>标签中若有 selected 属性，则表示该项默认为选中。

【例 5.4】　使用下拉框提供多选一的选项。

程序清单　文件 web\5\html\select1.html

```
1：<html>
2：<head>
3：   <title>下拉框示例一</title>
4：</head>
5：<body>
6：     <form>
7：        <select   name="city">
8：            <option>上海</option>
9：            <option>北京</option>
10：            <option>广州</option>
11：        </select>
12：       </form>
13：</body>
14：</html>
```

运行效果如图 5-7 所示。



图 5-7　下拉框的使用

在这个例子中，若选中选项“北京”，则下拉框 city 的值就取为“北京”这个字符串。在下例中，我们设置了 option 标签的 value 属性，当该选项被选中时，下拉框的取值即为该 value 属性的值。

【例 5.5】　设定 select 下拉框中各选项的取值。

主要代码　文件 web\5\html\select2.html

```
1:    <body>
2:        <form action="save.jsp">
3:            <select   name="city">
4:                <option   value="sh">上海</option>
5:                <option   value="bj">北京</option>
6:                <option value="gz">广州</option>
7:            </select>
8:            <input type="submit" value="确定"/>
9:        </form>
10:   </body>
```

此时，虽然页面显示效果和前例是相同的，但是当选中“上海”这个选项，然后单击提交按钮时，浏览器的地址中将出现如图 5-8 所示的结果。

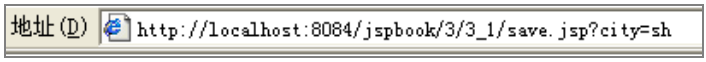


图 5-8　设置了 value 属性后的下拉选项值

地址栏中“?”后面的内容“city=sh”即为页面 select2.html 向 save.jsp 提交的数据，也称为参数，这里提交的参数名称为“city”，而值为“sh”。

可以在<option>标签中使用 selected 属性设定一组下拉菜单中的默认选项，格式如下：

```
<option   value="bj" selected>北京</option>
```

当<select>标签中含有 multiple 属性时，它表现为列表框的形式，可通过“Ctrl”键与鼠标左键组合选取多个选项。

3．超链接标签

该标签用于从一个网页跳转到另一个网页。

格式：

```
<a href="目标 URL" target="打开窗口方式" title="提示文字">
        浏览器中显示的热点
</a>
```

属性说明如下。

■ href：单击后要打开的目标资源（通常即网页）的路径，通常设为相对路径。

■ title：当鼠标移到链接上方时显示的提示文字。

■ target：表示目标资源的内容将在哪个位置打开，可取值如下。

- _blank：在新的浏览器窗口中打开。
- _parent：在父框架窗口中打开。
- _self：在当前窗口中显示目标 URL 中的内容，此为默认值。
- _top：将目标 URL 中的内容显示在最上层的窗口中。
- 框架名称：用在框架网页中，用于将目标 URL 中的内容显示在指定的框架中。

例如，登录，页面上显示的文字“登录”成为一个超链接，当单击这个超链接后，将向服务器请求 login.jsp 的页面内容。

目标资源的路径可使用绝对路径，也可使用相对于当前页面的相对路径。建议使用相对路径。

超链接中的热点可以是文字，也可以是图片等其他形式。

4．图片标签

这是一个单标签，用于显示指定的图片。格式如下：

```
<img src="images/logo.gif" width="200" alt="头像" />
```

■ src 属性指明要显示的图片文件的路径，与超链接一样，也可使用绝对路径或相对路径。

■ width 属性表示所显示的图片的宽度，这里设为了 200px，即 200 个像素。还可设定 height 属性，表示图片的显示高度。

■ alt 属性表示未找到相应的图片文件时，用以替代图片而显示的文字内容。

在网站开发中，网站的各类资源应分类存放，以利于网站的后期维护。例如，所使用到的各种图片，就应该单独存放在一个或多个文件夹中。

5.2.2　CSS 简介

在设计网页的原型时，为了使论坛的所有页面维持相同的风格，我们对每个页面采用了相同的布局，自上而下划分为页首、导航条、内容及页脚 4 个区域。HTML 的 div 标签可以表示页面上的一个区域，再结合 CSS（层叠样式表）对每个区域进行定位控制，很容易完成与原型一致的布局效果。

CSS（Cascading Style Sheet）是级联样式表或层叠样式表的简称，用来控制页面元素的字体、字号、色彩、背景、间距大小及位置等格式属性，从而制作出丰富多彩的网页效果。DIV+CSS 页面布局模式是 W3C 标准的一个典型应用，也是目前主流的页面布局模式。本小节首先介绍 CSS 的基本知识，之后介绍 DIV+CSS 模式的布局方法。

1．在网页中使用 CSS 的 3 种方式

按照 CSS 代码在网页文件中出现的不同位置，可以将 CSS 划分为内联样式、嵌入样式和外部样式 3 种形式。

1）内联样式

内联样式又称为“行内样式”，它可通过设定 HTML 标签的 style 属性值嵌入 CSS 代码。例如：

```
<p style="color:black">Hello</p>
```

2）嵌入样式

在页面的头部使用<style></style>标签对嵌入 CSS 代码。例如：

```
<head>
  <style>
    body{
      text-align:center;
      font-size:12px;
      color:#ccc;
    }
  </style>
<!--其他代码-->
</head>
```

3）外部样式

将 CSS 代码编写在一个独立的文件中，该文件的后缀名为.css，而后在页面文件中使用<link>标签或<style>...</style>标签对来指明所需要使用的样式文件。例如，CSS 文件名为 my.css，则在页面中引入该样式文件的代码为：

```
<link rel="stylesheet" href="my.css" style="text/css" />
```

或

```
<style type="text/css">
  <!--
    @import url("my.css");
  -->
</style>
```

这种形式使用最普遍，便于多个页面保持相同的风格。

2．CSS 选择器

CSS 通过为指定的元素定义样式来达到样式控制的目的。定义 CSS 样式的格式为：

```
选择器{
属性名称 1：属性值 1;
属性名称 2：属性值 2;
...
}
```

基本的选择器有以下 3 种。

1）标签选择器

通过 HTML 的标签名称来指定样式。例如，CSS 代码“p{color:blue;}”表示将所有的<p>标签中的文字设置为蓝色。

2）id 选择器

若为页面的一个元素指定了 id，则可通过 id 单独为其定义 CSS 样式。一般来说，元素的 id 在整个页面中应是唯一的，通过 id 定义 CSS 的语法格式如下：

```
#id 名称{
属性名称 1：属性值 1;
属性名称 2：属性值 2;
...;
属性名称 n：属性值 n;
}
```

注意 id 名称前面应有一个符号“#”。

【例 5.6】　id 选择器的使用。

```
1：<html>
2：<head>
3：    <style>
4：        #foot{width:100%;background-color:#ccc;color:#fff;}
5：</style>
6：</head>
7：<body>
8：<!--其他代码-->
9：<div id="foot">版权所有</div>
10：</body>
11：<html>
```

3）类选择器

若页面上的多个元素都要使用相同的样式，则可通过 class 属性为这些元素指定相同的类名，然后为该类定义一组样式。通过类名定义样式的语法格式如下：

```
.类名{属性名称：属性值;}
```

注意类名前有一个圆点符“.”。

【例 5.7】　CSS 类选择器的使用。

```
程序清单　web\5\css\poem.html
4：<html>
5：<head>
6：<title>类选择器的使用</title>
7：<link href="css/class.css" rel="stylesheet" type="text/css" />
8：</head>
9：<body>
10：    <p class="title">宿建德江</p>
11：    <p class="author" >孟浩然</p>
12：    <p class="content">移舟泊烟渚，日暮客愁新。</p>
13：    <p class="content">野旷天低树，江清月近人。</p>
14：    <hr/>
15：    <p class="title">江畔独步寻花</p>
16：    <p class="author" >杜甫</p>
17：    <p class="content">黄四娘家花满蹊，千朵万朵压枝低。</p>
18：    <p class="content">留连戏蝶时时舞，自在娇莺恰恰啼。</p></body>
19：</html>
```

程序清单　web\5\css\css\class.css

```
1：@charset "utf-8";
2：body{
3：text-align:center;
4：}
.5：title{
6：font-weight:bold;
7：font-size:14px;
8：}
9：.author{
10：color:red;
11：text-decoration:underline;
12：font-size:16px;
13：font-family:"隶书";
14：}
15：.content{
16：font-size:12px;
17：}
```

访问 poem.html，运行效果如图 5-9 所示。

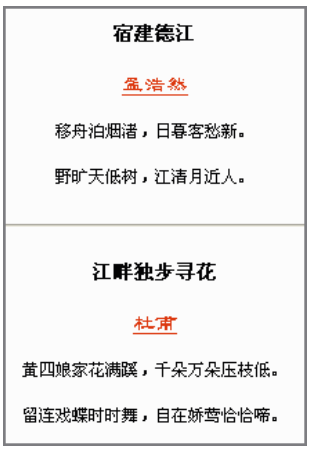


图 5-9　类选择器的使用

3．定位与浮动

div 标签表示页面上的一个区域，一个页面可使用多个 标签划分为不同的区域或层次。通过设定 div 的 float（浮动）属性或 position（定位）属性，可以确定 div 在页面上的显示位置。

【例 5.8】　三行两列式页面布局示例。

程序清单　文件 web\5\css\div.html

```
1：<html>
2：   <head>
3：          <meta http-equiv="Content-Type" content="text/html;
               charset=utf-8" />
```

```
4:  <title>三行两列式布局示例</title>
5:  <link href="css/main.css" rel="stylesheet" type="text/css" />
6:  </head>
7:  <body>
8:      <div id="wrap">
9:          <div id="head">页首区域</div>
10:             <div id="nav">
11:                 导航条区域
12:             </div>
13:             <div id="content">
14:                 内容区域
15:             </div>
16:         <div id="foot">页脚区域</div>
17:     </div>
18:  </body>
19: </html>
```

第 5 行表示载入层叠样式表文件 css\main.css，该文件对各层的大小、背景色、位置等进行了定义，内容如下：

程序清单　文件 web\5\css\css\main.css

```
1:  @charset "utf-8";
2:  *{
3:      border :1px  solid  #000;
4:  }
5:  body{
6:      font-family: "宋体";
7:      font-size: 14px;
8:      text-align: center;
9:      margin:0;
10: }
11: #wrap {
12:     width:75%;
13:     margin: 0 auto;
14: }
15: #head {
16:     width:100%;
17:     height:120px;
18:     background-color: #999;
19:     font-family: "方正舒体";
20:     font-size: 48px;
21:     font-style: oblique;
22:     line-height:120px;
23:     color:#D5D5D5
24: }
25: #nav {
26:     width:21.5%;
```

```
27：        height:400px;
28：        float:left;
29：        background-color: #FF9900;
30：        padding-top:20px;
31：        text-align:center;
32：  }
33：  #content {
34：        float: right;
35：        width: 78.3%;
36：        padding-top: 10px;
37：        background-color: #FFF09D;
38：        height:400px;
39：  }
40：  #foot {
41：        clear: both;
42：        height: 40px;
43：        width: 100%;
44：        background-color: #999;
45：        padding-top:10px;
46：  }
```

页面运行效果如图 5-10 所示。



图 5-10　常见的三行两列式布局

这里为了演示布局的效果，设置了内容区域的高度，实际使用时，也可不设定具体的高度，由浏览器根据内容的多少自动调整。

只需修改相关的 CSS 代码，即可得到与 5.1 节相同的论坛网页的布局效果。

5.2.3　使用 HTML＋CSS 制作注册页面

在学习了前面两个小节的知识后，借助于静态网页制作工具 Dreamweaver，不难制作出论坛的用户注册页面，代码如下：

程序清单　文件 web/5/5_2/reg.jsp

```
1：<html>
2：      <head>
```

```
3:          <meta http-equiv="Content-Type" content="text/html;
             charset=UTF-8">
4:          <title>学习论坛--注册 </title>
5:          <style type="text/css">
6:              <!--
7:              @import url("css/common.css");
8:              -->
9:          </style>
10:     </head>
11:  <body id="reg">
12: <div id="container">
13:  <div id="logo">
14:    <img src="images/slogan.gif" />
15: </div>
16: <!--层 nav，用来显示导航条，导航条用列表的形式列出来 -->
17: <div id="nav">
18:     <div id="welcome">
19:         游客,欢迎进入 JSP 学习论坛
20:     </div>
21:     <ul>
22:         <li><a href="index.jsp">首页</a></li>
23:         <li><a href="login.jsp">登录</a></li>
24:         <li><a href="reg.jsp">注册</a></li>
25:         <li><a href="mailto:sunnyworld126@126.com">与我联系</a></li>
26:     </ul>
27: </div><!--nav 层结束-->
28: <div id="content">
29: <form action="chkReg.jsp" method="post">
30:       <!--表单的各项内容用表格排列整齐-->
31:       <table>
32:           <tr><td colspan="2" id="title">请填写注册信息</td></tr>
33:             <tr><td class="ltd"> 用户名* </td>
34:               <td>
35:             <input type="text" name="uname" size="40" maxlength="20" />
36:               </td>
37:           </tr>
38:           <tr><td class="ltd"> 密码* </td>
39:               <td>
40:               <input type="password" name="upw" size="40" maxlength="20"/>
41:               </td>
42:           </tr>
43:           <tr><td class="ltd"> 确认密码* </td>
44:               <td>
45:           <input type="password" name="upw2" size="40" maxlength="20"/>
46:               </td>
47:           </tr>
```

```
48:              <tr><td class="ltd">  性别  </td>
49:                   <td>
50:                    <input type="radio"    name="usex" value="男" />男
51:                    <input type="radio" name="usex" value="女" />女
52:                    </td>
53:                </tr>
54:                <tr>
55:                   <td class="ltd">Email</td>
56:                   <td><input type="text" name="uemail" size="40"></td>
57:              </tr>
58:              <tr><td class="ltd">选择头像</td>
59:                  <td style="vertical-align:middle">
60:                   <input type="hidden" name="ulogo" id="ulogo" />
61:                   <img src="images/head/space_noface.gif" id="hlogo"
                       name="hlogo" width="70"/><br>
62:                   <input type="radio" name="s1" />选择
63:                   <input type="radio" name="s1" />上传
64:                    </td>
65:              </tr>
66:              <tr>
67:              <td> </td>
68:              <td align="center">
69:                 <input type="submit" value="确定" /> 
70:                 <input type="reset" value="重置" />
71:              </td>
72:                 </tr>
73:                 </tfoot>
74:            </table>
75:         </form>
76:       </div><!--content 层结束-->
77:        <div id="footer">
78:        2008-2010 &copy; V-Sunny Technology Company. All rights reserved.
79:        </div>
80:     </div> <!--container 层结束-->
81:  </body>
82: </html>
```

为了节省篇幅，这里不再列出 CSS 源码，有需要的读者请自行参考本书所提供的案例源码。将文件 reg.jsp 复制到项目的 web 目录下，访问该页面，得到如图 5-11 所示的访问效果。

这里要重点提示以下 3 点。

- form 标签的 action 属性表示接收表单数据的网页名称，form 标签的 method 属性表示提交数据的方式，可取“post”或“get”，默认值为“get”。建议设为“post”方式。
- input 标签的 type 属性取不同的值，就会在页面上表现为不同形态的输入元素。当 type 取值为“hidden”时，不在页面上显示。
- 通常采用 DIV+CSS 的方式完成网页的布局，为使不同的网页保持一致的风格，可编写一个通用的.css 文件，在不同的网页中引用该样式表文件即可。



图 5-11　注册页面运行效果图

5.2.4　JavaScript 简介

JavaScript（简称 JS）是由 Netscape 公司推出的客户端脚本语言，用 JavaScript 编写的代码是由浏览器来解释执行的，主要应用于制作网页特效、客户端数据验证等等，目前基本上所有的浏览器都内嵌了对 JavaScript 的支持。近年来，随着 Ajax 技术的兴起，JavaScript 又在 Web 开发领域重现了生机。这里只简单介绍利用 JavaScript 实现网页的一些动态效果及表单数据的验证的方法。

1．JavaScript 的基本语法

通过比较 JS 与 Java 语法风格上的异同，有助于快速掌握 JavaScript 的基本语法，比较结果如下。

- 与 Java 一样，JS 是对大小写敏感的。
- JS 的分支语句、循环语句格式与 Java 基本相同。
- JS 中的语句不必以分号结尾，但为了与 Java 保持一致，建议加上分号。
- JS 中的字符串可用双引号括起，也可用单引号括起。
- JS 中的变量定义采用“var s;”这种形式，即不指出该变量的类型，其类型由所赋的值决定。一个变量甚至可以不定义就直接使用。
- JS 中的函数定义以关键字 function 引导，参数及返回值的类型不指出。

2．JS 的事件处理

JS 主要通过其事件处理机制完成各种功能。除了定时器事件外，主要是针对用户的键盘输入或鼠标动作而进行的处理，如 click（鼠标单击）事件、mouseover（鼠标移上）事件、mouseout

（鼠标移开）事件、focus（得到焦点）事件、blur（失去焦点）事件、submit（提交表单）事件等。若需要对发生在一个页面元素上的某一事件进行处理，只需在该元素的开始标签中添加 onXXX 属性，其中的“XXX”为事件的名称，该属性的值即为事件发生时的响应代码。

【例 5.9】　JS 函数的定义与调用。

程序清单　文件 web\5\js\js_func.html

```
1：<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2：<html>
3：      <head>
4：           <title>JS----函数定义与调用</title>
5：           <meta http-equiv="Content-Type" content="text/html;
               charset=UTF-8"/>
6：           <script>
7：                function to(name)
8：                {
9：                     alert("Hello,"+name);
10：                }
11：           </script>
12：      </head>
13：      <body>
14：           <input type="button" value="请点击" name="button1"
                    onclick="to('三毛')"/>
15：      </body>
16：</html>
```

代码说明如下。

- 该页面中，第 7～10 行代码定义了一个函数 to，需要一个参数 name。该函数的作用是弹出一个警告框，框内显示文字“Hello,”及 name 参数的内容。
- 第 14 行通过代码“onclick="to('三毛')"”指明，当单击了页面上的“请点击”按钮后，将调用 to 函数，传递的参数是“三毛”。因此，单击按钮后将弹出警告框，其中显示文字“Hello,三毛”。

3．在页面中加入 JS 代码的方式

上例采用了直接在<script>...</script>标签之间编写 JS 代码的方式，除了这种方式外，还可以采用以下两种方式在页面中加入 JS 代码。

- 直接写在 HTML 标签的事件处理属性中。

例如，当输入框得到输入焦点时，将框内的文字清空：

```
<input type="text" value="请在此输入用户名" onfocus="this.value=" " />
```

- 将 JS 代码编写成独立的 JS 文件，在<head></head>部分利用<script>标签导入。

【例 5.10】　外部 JS 文件的引用。

将 JS 代码单独保存为 myjs.js，而后在 welcome.jsp 中引入。这里新建一个子目录 js，专门用于存放 js 文件。

程序清单　文件 web\5\js\js\myjs.js

```
1：function welcome()
```

```
2：{
3：      alert("欢迎来到 JSP 学习天地！");
4：}
```

程序清单　文件 web\5\js\welcome.jsp

```
1：<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2：<html>
3：   <head>
4：      <!--其他代码略-->
5：      <script src="js/myjs.js">
6：      </script>
7：   </head>
8：   <body onload="welcome()">
9：         This is my html page.
10：  </body>
11：</html>
```

4．实现网页的动态效果

为了能控制页面上的元素，首先要能找到这个元素。一般通过两个步骤实现。

（1）为元素指定一个 id，该 id 在整个页面中是唯一的。

（2）利用 JS 代码“document.getElementById(id)”，根据 id 找到这个元素。

许多网站在实现用户注册功能时，一般只要求用户输入一些必填信息，而一些高级选项则由用户自行决定是否填写。这时，通过单击页面上的“高级选项”复选框，可以显示或隐藏网页上的高级选项部分的内容。下面的例子模拟了这一功能。

【例 5.11】　打开/关闭注册表单的高级选项。

程序清单　文件 web\5\js\advToggle.html

```
1：<html>
2：    <head>
3：        <meta http-equiv="Content-Type" content="text/html;
             charset=UTF-8">
4：        <title>打开/关闭高级选项</title>
5：        <script>
6：            function toggle()
7：            {
8：                var adv=document.getElementById("adv");
9：                if (adv.style.display=="block")
10：                    adv.style.display="none";
11：                else
12：                    adv.style.display="block";
13：            }
14：        </script>
15：    </head>
16：    <body>
17：        <form action="">
18：            请填写基本信息：
```

```
19：                <div id="base" style="background-color:#eee;">
20：                    用户名:<input type="text" name="name" /><br/>
21：                    密码：<input type="password" name="pw" />
22：                </div>
23：                高级选项<input type="checkbox" onclick="toggle()"/><br/>
24：                <div id="adv" style="display:none;">
25：                    通信地址<input type="text" name="addr" /><br/>
26：                    联系电话<input type="text" name="cell" />
27：                </div>
28：                <input type="submit" value="提交" />
29：            </form>
30：        </body>
31：</html>
```

这里主要通过 JS 改变了高级选项层的 display 属性值，当该属性取“none”值时，该元素在页面上不显示，而将其设为“block”或“inline”时，它会以块框或行内框的形式显示在页面上。

该页面关闭与打开“高级选项”时的效果如图 5-12 所示。

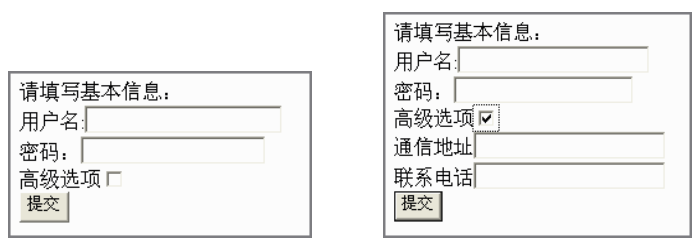


图 5-12　关闭/打开“高级选项”的注册页面

5.2.5　为注册表单添加 JavaScript 验证

当用户在表单中单击了“提交”按钮后，则发生提交事件。通过对表单的提交事件进行处理，可以实现对表单数据的验证。由于 JS 代码是由浏览器解释执行的，因此用 JS 实现的数据验证也被称为“客户端验证”或“前端验证”。这时表单的一般形式为：

```
<form action="next.jsp" onsubmit="return check()">
    ...
    <input type="submit" value="提交" />
</form>
```

具体的验证工作则在函数 check 中进行。当 check 函数的返回值为 false 时，提交动作被阻止，否则提交动作将继续。这里应注意，onsubmit 属性的值是“return check()”，而非“check()”，后面这种写法不能起到拦截提交动作的作用。

1．获取输入元素的值

一般地，在 JavaScript 中，元素的 value 属性表示该元素的输入值，因此，若有一个元素的 id 为“uname”，则获取该元素值的表达式是“document.getElementById("uname").value”，该表达式的值是字符串类型。为了简化代码，常定义一个基础函数，完成根据 id 寻找页面元素的功能，如下所示：

```
function $(id){
    obj=document.getElementById(id);
    return obj;
}
```

此时表达式“document.getElementById("uname").value”可简化为“$("uname").value”。

2．判断输入值是否为空或空串

表单中有些数据是必须填写的，即这些输入值不能为 null 或空字符串。

【例 5.12】　检测输入的用户名是否为空。

程序清单　　文件 web\5\js\emptyCheck.html

```
1：<html>
2：<head>
3：<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
4：<title> 客户端信息检测--内容是否为空 </title>
5：<script>
6：    /*   通用函数--获取指定 id 的 object*/
7：    function $(id){
8：        obj=document.getElementById(id);
9：        return obj;
10：    }
11：    /*检测函数 check，检测用户名是否为空*/
12：    function check()
13：    {
14：        var name=$('uname').value;
15：        if (name==null || name=='')
16：        {
17：            alert('请输入用户名');
18：            return false;
19：            }
20：        else
21：            return true;
22：        }
23：</script>
24：</head>
25：<body>
26：        <form name="frm1" action="next.jsp" onsubmit="return check()" >
27：            用户名<input type="text" name="uname"    id="uname" />
28：            <input type="submit" value="提交"/>
29：        </form>
30：</body>
31：</html>
```

代码说明如下。

- 第 27 行代码指出输入框的 id 为“uname”。
- 第 26 行代码指出，当用户提交表单时，需先调用 check 函数，若该函数返回值为 false，

则不予提交，否则将数据提交给 action 属性指明的页面 next.jsp。

- 代码第 12～22 行定义了检测函数 check，第 14 行代码取出文本框 uname 的输入内容，并将值赋给变量 name。代码第 15～21 行判断变量 name 的值是否为空。若是则弹出警告框，并返回 false 值（提交动作被拦截）；若非空，则返回 true 值，允许继续提交给页面 next.jsp。

3．判断字符串长度是否在指定范围内

在 JavaScript 中，字符串对象的 length 属性表示字符串的长度值，即字符的个数，实际处理时可能需要计算的是字节数。因此，当字符串中包含中文或其他双字节字符时，使用 length 属性不能得到相应的字节数，这时可通过下面的函数来计算字符串的字节数：

```
String.prototype.chLength=function(){
    var len = this.replace(/[^\x00-\xff]/g, "**").length;
    return len;
    }
```

这段代码为所有的字符串对象添加一个新的自定义函数 chLength，该函数利用正则表达式去匹配串中的所有双字节字符，将每个双字节字符替换为 2 个“*"号，然后计算转换后的字符串的字符长度。

若有字符串变量 s，则表达式 s.chLength()的值即为它的字节长度（双字节字符被计为 2 个字节）。

【例 5.13】　判断输入的用户名的长度是否在指定范围内。

程序清单　文件 web\5\js\lengthCheck.html

```
1：<html>
2：<head>
3：<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
4：<title> 客户端信息检测--输入内容的长度是否符合要求 </title>
5：<script>
6：     /*   通用函数--获取指定 id 的 object*/
7：     function $(id){
8：         obj=document.getElementById(id);
9：         return obj;
10：    }
11：    /*通用函数----返回字符串的长度，1 个汉字计为 2 个字节*/
12：    String.prototype.chLength=function(){
13：        var len = this.replace(/[^\x00-\xff]/g, "**").length;
14：        return len;
15：        }
16：    /*检测函数 check，检测用户名的长度是否在指定范围内*/
17：    function check()
18：    {
19：        var name=$('uname').value;
20：        if (name.chLength()>16|| name.chLength()<3)
21：        {
22：            alert('用户名长度应在 3-16 之间');
```

```
23：                return false;
24：                }
25：            else
26：                return true;
27：      }
28：</script>
29：</head>
30：<body onload="$('uname').focus()"    >
31：      <form name="frm1" action="next.jsp" onsubmit="return check()" >
32：          用户名<input type="text" name="uname"    id="uname" />
33：          <input type="submit" value="提交"/>
34：      </form>
35：</body>
36：</html>
```

该例与前一个例子的处理方法基本相同，只是具体判定的条件不同。第 20 行代码判断所输入的内容的长度是否符合要求，是则返回 true 值，否则返回 false 值。

4．判断字符串是否为指定的格式

像 E-mail 地址、电话号码这样的数据，一般要求符合某种特定的格式，结合正则表达式，很容易完成类似的验证处理。

在 JavaScript 中，通过正则表达式对象 RegExp 的 test 函数可以判断一个字符串是否与指定的模式相匹配。

【例 5.14】　判断输入的 E-mail 格式是否正确。

程序清单　文件 web\5\js\formatCheck.html

```
1：<html>
2：<head>
3：<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
4：<title>  客户端信息检测--输入内容的长度是否符合要求  </title>
5：<script>
6：      function $(id){
7：          obj=document.getElementById(id);
8：          return obj;
9：      }
10：     /*chkEmail()---检测 E-mail 是否符合规定的格式*/
11：      function chkEmail(){
12：         //定义表示 email 格式的正则表达式
13：         var pattern =    /^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$/;
14：         var email = $('email').value;
15：         //判断 email 输入是否为空或不符合格式要求
16：         if (email=="" || !pattern.test(email))
17：            {    alert("请填写正确的 Email.");     //弹出警告框
18：                 return false;              // 验证结果为 false，即不通过
19：            }
20：          else
21：                 return true;   // 验证结果为 true，表示通过验证
```

```
22：        }
23：</script>
24：</head>
25：<body>
26：    <form action="next.jsp" onsubmit="return chkEmail()" >
27：        Email 地址<input type="text" name="email"    id="email" />
28：        <input type="submit" value="提交"/>
29：    </form>
30：</body>
31：</html>
```

代码说明如下。

- 代码第 13 行定义了匹配 E-mail 格式的模式字符串 pattern（正则表达式）。
- 第 16 行的“pattern.test(email)”用于检测字符串 email 的内容是否符合 pattern 所指明的格式，若符合，则 test 函数返回 true 值，否则该返回 false 值。
- 其余代码与例 5.12 相似，请参考前面的内容。

当需要验证输入的内容是否符合指定的格式时，使用正则表达式可以大大简化我们的代码编写工作。

下面是几个常用的正则表达式：

```
中文：[\u4e00-\u9fa5]
双字节字符(含中文)：[^\x00-\xff]
座机电话号码：/^\d{3}-\d{8}|\d{4}-\d{7}$/
国内邮政编码：/^[1-9]\d{5}$/
整数：/^-?[1-9]\d*$/
```

当然，一个正则表达式是否适用，还需要根据具体情况进行调整。

5．综合验证

上面的例子展示了对输入内容进行单项检测的情形，而实际情况往往是需要对多项数据进行检测。这时，可以为每项需要检测的数据设置一个有效标志 flag，若该数据验证通过，则将 flag 设为 true，否则设为 false。最后检测是否所有待检测数据的有效标志均为 true，是则允许将表单提交到服务器，否则返回 false 不提交到服务器，并在页面上显示错误提示信息。

【例 5.15】　注册页面的客户端验证。

为了节省篇幅，这里的注册输入页面只保留主要的表单部分代码，有兴趣的读者可以根据此例修改 5.2.3 小节中的 reg.jsp 文件，使之具备客户端验证的功能。

程序清单　文件 web\5\js\reg.html 的文件内容：

```
1：<html>
2：    <head>
3：        <meta http-equiv="Content-Type" content="text/html;
           charset=UTF-8">
4：        <title>表单数据综合检测</title>
5：        <script src="js/check.js">
6：        </script>
7：    </head>
```

```
8：      <body>
9：          请填写：
10：          <div id="msg" style="color:red"></div>
11：          <form name="frm1" action="chkReg.jsp" onsubmit="return chkAll()">
12：          用户名<input type="text" name="uname"    id="uname"    /><br/>
13：          密码<input type="password" name="pw1" id="pw1" /><br/>
14：          确认密码<input type="password" name="pw2" id="pw2" /><br/>
15：          Email <input type="text" name="email" id="email"    /> <br/>
16：              <input type="submit" value="提交" name="mysubmit"/>
17：              <input type="reset" onclick="frm1.reset();init();"/>
18：          </form>
19：      </body>
20：</html>
```

check.js 中定义了数据验证的各个函数，代码如下：

```
程序清单　文件 web\5\js\js\check.js
1：/*定义表示出错信息的字符串变量*/
2：var err='';
3：/*下面是表示各输入项是否有效的标志变量，初始值为 false*/
4：var name_valid=false;
5：var pw_valid=false;
6：var email_valid=false;
7：/*----------------下面是几个通用的函数----------------*/
8：/*　通用函数--获取指定 id 的 object*/
9：function $(id){
10：      obj=document.getElementById(id);
11：      if (obj==null) obj=document.all.id;
12：      return obj;
13：}
14：/*通用函数----返回字符串的长度，1 个汉字算 2 个字符*/
15：String.prototype.chLength=function(){
16：      var len = this.replace(/[^\x00-\xff]/g, "**").length;
17：      return len;
18：}
19：/*初始化函数，在按了 reset 键时调用*/
20：function init()
21：{
22：      $('uname').focus(); //定位在第一个输入框
23：      $('msg').innerHTML="";     //清除错误信息
24：      err='';
25：}
26：/*函数 chkName---检查用户名的有效性*/
27：function chkName()
28：{
29：      name_valid=false;
30：      var name=$('uname').value;   //获取所输入的用户名
```

```
31:     //检测用户名是否为空
32:     if (name==null||name=='')
33:     {
34:         err+='用户名不能为空.<br/>';
35:     }
36:     else
37: //检测用户名的长度是否为 3-16 个字符，汉字按 2 个字符计算
38:     if(name.chLength()>16||name.chLength()<3)
39:     {
40:         err+='用户名的有效长度在 3-16 之间.<br/>';
41:     }
42:     else
43:     {
44:         name_valid=true;
45:     }
46: }
47: /*函数 chkPassword()--检测密码是否有效*/
48: function chkPassword()
49: {
50:     pw_valid=false;
51:     var pw1=$('pw1').value;
52:     var pw2=$('pw2').value;
53:     if (pw1!=pw2)
54:     {
55:         err+='两次输入的密码不一致. <br/>';
56:     }
57:     else
58:      {
59:         var pattern=/^([\w*%$-_]){3,12}$/;
60:         if (!pattern.test(pw1))
61:         {
62:             err+='密码中只能由 3-12 位字母、数字或符号%、*、$、-、_组成.<br/>';
63:         }
64:         else
65:             pw_valid=true;
66:     }
67: }
68: /*chkEmail()---检测 email 是否符合规定的格式*/
69: function chkEmail(){
70:     email_valid=false;        //设检测标志为不通过
71:    //定义表示 email 格式的正则表达式
72:     var pattern = /^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$/;
73:     var email = $('email').value;
74:     if (email=="" || !pattern.test(email))
75:     {//email 输入为空或不符合格式要求
76:         err+='Email  地址无效，请重新填写。<br/>';
```

```
77:         }
78:         else
79:         {
80:             email_valid=true;
81:         }
82: }
83:
84: /*-----检测是否所有的输入项都合法，是则允许提交数据------*/
85: function chkAll()
86: {
87:         $('msg').innerHTML="";
88:         err='';
89:         chkName();
90:         chkPassword();
91:         chkEmail();
92:         if (name_valid && pw_valid && email_valid)
93:             return true;
94:         else
95:         {
96:             $('msg').innerHTML=err;     //显示出错信息
97:             return false;
98:         }
99: }
```

当密码的长度不符合要求或 Email 地址不符合格式要求时，验证结果如图 5-13 所示。

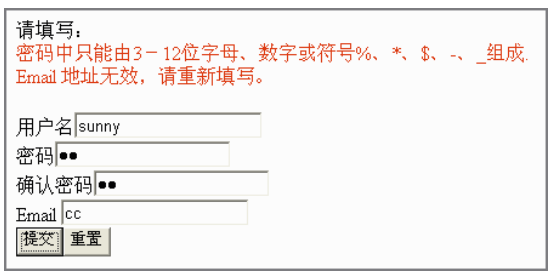


图 5-13　利用 JS 完成表单数据的验证

NetBeans 中提供了对 JavaScript 代码的调试功能，调试方法请参考第 3 章相关章节的内容。

此外，还可以利用一些现成的 JS 验证框架帮助我们实现一般的客户端验证工作。例如，由中国人自己开发的免费框架 JSValidation，就是一个比较优秀的验证框架。它的使用也比较简便，有兴趣的读者可以到其官方网站上下载，网址是 http://cosoft.org.cn/projects/jsvalidation。



思考与训练

一、基础训练（请按要求写出相应的 HTML 标签）

1．名称为 reg 的表单，提交到 chk.jsp 处理，提交方式为 post，请写出相应的表单标签。

2．名称为 uname 的单行文本框，在框前面显示的文字为“用户名”。

3．名称为 pw 的密码输入框，框前面的文字为“密　码”，“密”和“码”中间有 2 个空格。

4．名称为 price 的单行文本框，框前面有文字“单价”，初始值为 0.0。

5．单行文本框，名称为 bookname，显示的长度为 60，最大允许的输入字符数为 50。

6．如下图所示的单选按钮，名称为“age”，默认的选项是“90 后”。



7．在上题中，为每个选项设置相应的 value 值，分别为 9、8、0。

8．如下图所示的性别选项，名称为“gender”，默认选项是第 1 个，对应的取值依次是“male”和“female”。



9．用下拉框提供 6 个景点，由用户选择想去的旅游景点，只能单选。

10．编写注释，注释的内容为“文件名：reg.jsp, 2009-3-5”。

11．提交按钮，按钮上的文字为“同意”。

12．隐藏域，名称为 pageno，值为 1。

13．图像链接，单击图片后打开的网页为“hotNews.jsp?id=2”。

14．超链接，显示的文字为“热点新闻”，单击后打开的页面为“hotNews.html”。

15．制作值班表，如下图所示。

时间		值班人员
周一	7:00-16:00	张军
	16:00-23:00	周建

16．提供多行文本框，输入用户的反馈意见。

二、思考题

1．<form>标签中的 method 属性可以取哪些值？有何不同的含义？

2．<form>标签中的 action 属性的作用是什么？

3．单选按钮和复选框分别适用于什么场合？

4．隐藏域是否会在页面上显示出来？一般用途是什么？

三、综合实训

1．编写对学校食堂服务的意见调查表，表单中应出现文本输入框、单选按钮、复选框、下拉框及多行文本框，使用 JavaScript 对其中的必填项等进行验证。

2．编写发表新帖的输入页面，使用 JavaScript 验证输入内容是否少于 4 个字符，若是，则弹出警告框，显示信息“请勿灌水”。

第 6 章　注册功能的实现

6.1　任务的提出与分解

在第 5 章中已经制作了注册信息的输入页面，那么，当新用户填写了合法的数据并提交到服务器后，还应进行哪些处理才能完成注册功能呢？如图 6-1 所示描述了注册业务的基本处理流程。

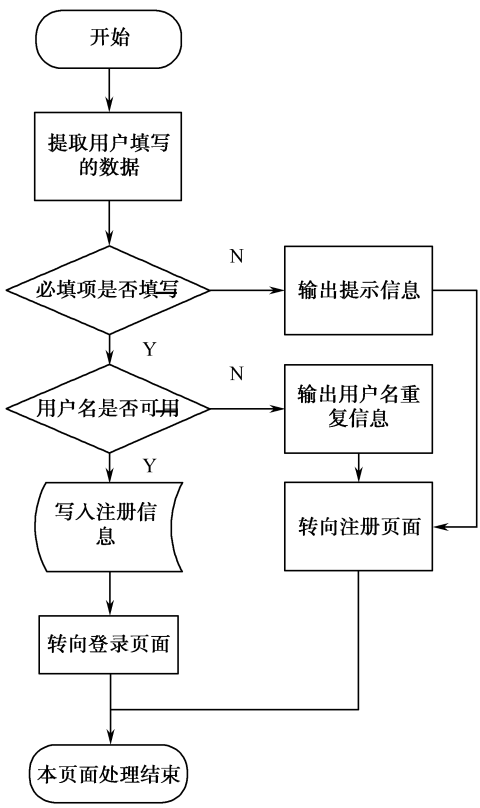


图 6-1　注册处理流程

从图中可以看出，首要任务是读取客户端提交的数据，其次是检测用户名是否已被人注册，若无人使用，则可将所填写的注册信息写入数据库的用户信息表中。当然，还应根据处理的结果向用户返回提示信息。可将此分解为以下几个基本的小任务。

- 提取来自客户端的数据。
- 向客户端输出信息。
- 查询数据库，检测用户名的可用性。
- 向数据库中写入数据。
- 页面转向控制。

下面让我们逐个学习相关的知识，分步完成上述的任务。

6.2　输出信息及提取表单数据

6.2.1　使用 page 指令设置页面编码

当我们在 NetBeans IDE 中新建一个 JSP 文件时，将得到一个根据 JSP 模板自动生成的页面文件，该文件的第一行代码就是一条 page（页面）指令。

JSP 页面中形如<%@ ... %>格式的语句被称为指令，当 JSP 引擎将 JSP 文件转换为 Java 文件时，这些指令将影响引擎的转换处理工作。

page 指令是 JSP 页面中使用频率最高的指令，主要用来设置整个 JSP 页面的相关属性，如网页内容的类型、网页的编码方式、网页需导入的包或类、发生异常时的处理方式等信息。它的语法格式为：

```
<%@ page  属性名 1="属性值 1" ... 属性名 n="属性值 n" %>
```

当我们的网页中含有中文信息时，应告知引擎对网页的内容进行编码输出时应采用哪个字符集。

page 指令一共有 13 个属性可以设置，详细内容请参考本书第 17.2.1 节的相关内容。要设置页面的编码字符集，可分别使用下面的 2 个属性来设置。

- contentType="页面 MIME 类型[;charset=字符集名称]"
- pageEncoding="字符集名称"

上面语法格式中的中括号表示括起的内容是可选的。

一般地，当网页中含有中文信息时，可将页面编码的字符集设为 GBK、GB2312 或 UTF-8。在本书中统一设为 UTF-8，如下所示：

```
<%@ page pageEncoding="UTF-8" %>
```

或

```
<%@ page contentType="text/html;charset=UTF-8" %>
```

6.2.2　使用 out 对象输出信息

out（输出）对象是 JSP 中最常用的内置对象之一。

1．什么是内置对象

JSP 中有一组内置的对象，即在 JSP 页面文件中不需创建就可直接使用的对象，它们提供了网络编程中经常使用的一些基本功能，如向浏览器输出信息，获取来自客户端的信息，向客户端发送响应信息，在客户端与服务器的会话期间存取信息等。这些内置对象在 JSP 网络程序设计中会起到非常强大的作用。

学过 Java 后，大家都知道，一个对象应先创建才能使用，为什么 JSP 中的这些对象不需创建就可使用呢？实际上，当一个 JSP 页面文件被转换为相应的 Servlet（一个特殊的 Java 类）文件时，Servlet 中就自动添加了这些对象的定义语句，通过查看该 Servlet 文件的代码，就不难理解其中的原因了。

2．out 对象

使用内置对象 out 可以向客户端输出信息。默认情况下，这些输出内容先写到一个输出缓冲区中，当发生下面 3 种情况之一时，才会将缓冲区中的内容送往客户端。

- 该页面所有的信息都已输出到缓冲区中。
- 输出缓冲区已满。
- 遇到“out.flush();”或“response.flushBuffer();”语句。

缓冲区的大小可使用 page 指令的 buffer 属性设置。

通常可使用 out 对象的 print()和 println()方法把内容写入输出缓冲区。例如：

```
<%   out.println("hello");
     out.print("Happy new year.");
     %>
```

输出缓冲区的内容就是即将发送到客户端的 HTML 格式的网页内容。与 print 方法相比，out 对象的 println 方法将使得网页的 HTML 代码中多一个换行，但在浏览器中显示时，HTML 代码中的换行符仅表现为一个空白符，并非换行。如果需要在浏览器中输出一个换行符，则应输出 HTML 中的换行标签
，即在需要换行的地方加入代码“out.print("
");”。

out 对象的 print 方法的参数类型与 System.out 的 print 方法基本相同，如基本数据类型的数据和 Object 类型的数据。但前者输出的内容最终是传送到客户端去的，而后者则将输出内容送往服务器的标准输出流，如控制台。

6.2.3　使用 request 对象读取请求参数

request（请求）对象是 JSP 的内建对象，封装了来自客户端的请求信息，使用 request 对象可以访问客户端发送给服务器的请求信息。在 JSP 程序设计中，request 对象的使用频率是非常高的。

1．request 对象

当客户端向服务器发起一个请求时，通常会把自身的一些信息，如用户填写的表单数据、保存在客户端的 Cookie 信息、客户端的 IP 地址等一起发送给服务器。服务器将来自客户端的请求信息封装到 request 对象中，而被请求的页面就可以使用 request 对象来获取这些信息了。封装在请求中的信息通常简称为“请求信息”，客户端发送给所请求页面的表单数据等通常被称为“请求参数”（parameter）。

request 对象的方法很多，要提取所填写的注册信息，可使用下面的方法：

```
String getParameter(String name)      //获取客户端发送的请求参数的值
```

该方法用于获取客户端发送给服务器的某参数的值，该参数的名称由 name 指出，返回值为 String 类型。

【例 6.1】　利用 request 对象提取表单数据。

在页面 inputName.jsp 中输入用户名后，提交到 printName.jsp，在页面 printName.jsp 中提取用户名并输出。

主要代码　文件 web\6\6_2\inputName.jsp

```
1：<body>
2：     <form action="printName.jsp" method="post">
```

```
3：　　用户名<input type="text" name="uname"/>
4：　　<input type="submit" value="提交" />
5：　</form>
6：</body>
```

当用户在该页面单击了“提交”按钮时，实际上是向服务器上的 printName.jsp 页面发起了请求，同时将表单中的数据（即用户名）等其他信息一起发送给 printName.jsp 页面。

主要代码　文件 web\6\6_2\printName.jsp

```
1：<%
2：String s;
3：s=request.getParameter("uname");
4：out.print("您输入的用户名是："+s);
5：%>
```

在页面中输入 Jerry，运行效果如图 6-2 和图 6-3 所示。

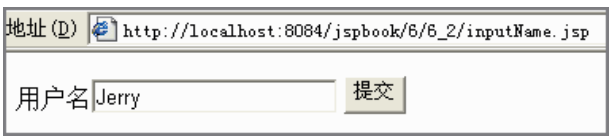


图 6-2　输入用户名

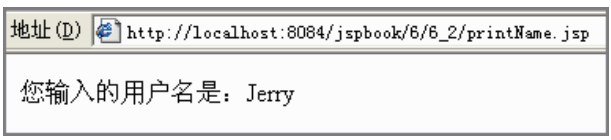


图 6-3　输出所填写的数据

让我们输入中文数据“姚明”再试试，结果得到如图 6-4 所示的“外星文”了！

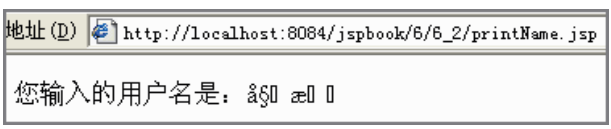


图 6-4　读取的中文参数为乱码

原来，若客户端通过 post 方式传送中文数据，则在提取数据前需正确设置请求信息编码时所使用的字符集，方法为：

```
void setCharacterEncoding(String charset)
```

该方法的参数即为字符集的名称。一般可设为 GB2312、GBK 或 UTF-8，建议设为 UTF-8。在页面 printName.jsp 的代码“s=request.getParameter("uname");”前添加下面的代码：

```
request.setCharacterEncoding("utf-8");
```

再次输入中文数据“姚明”，取出的数据如图 6-5 所示。



图 6-5　正确显示的中文参数

2．实验与思考

将 inputName.jsp 中第 3 行代码改为“用户名<input type="text" name="username"/>”，再次访问 inputName.jsp 并提交，将得到什么结果？为什么？应如何修改 printName.jsp？

6.2.4　任务实现

有了前面的知识，读取新用户填写的注册信息就易如反掌了。代码如下所示：

程序清单　web\6\6_2\chkReg.jsp

```
1：<%@page contentType="text/html" pageEncoding="UTF-8"%>
2：<html>
3：<head>
4：<title>读取用户填写的注册信息</title>
5：<meta http-equiv="contentType" content="text/html;charset=utf-8" />
6：</head>
7：<body>
8：<%
9：String uname,upw,upw2,usex,uemail,ulogo;
10：request.setCharacterEncoding("utf-8");
11：uname=request.getParameter("uname");
12：upw=request.getParameter("upw");
13：upw2=request.getParameter("upw2");
14：usex=request.getParameter("usex");
15：uemail=request.getParameter("uemail");
16：ulogo=request.getParameter("ulogo");
17：if (uname==null||uname.equals("")||upw==null||upw.equals(""))
18：{
19：      out.print("请填写用户名和密码.");
20：}
21：else if (!upw.equals(upw2))
22：      out.print("两次输入的密码不一致!");
23：else {
24：      out.print("用户名:"+uname);
25：      out.print("密码 1:"+upw);
26：}
27：%>
28：</body>
29：</html>
```

代码说明如下。

■ 由于用户填写的用户名可能是中文数据，因此必须在读取数据前加上第 10 行的代码，

将请求的编码字符集设为 utf-8。

- 第 11～16 行代码取出了来自注册页面的 6 项数据。
- 第 17～22 行对数据的正确性进行了基本的检测，虽然在客户端可能已经使用 JS 代码对数据进行了前端验证，但由于 JS 代码可能会被用户屏蔽，因此在服务器端仍需做基本的数据验证。

为了验证是否取到了正确的数据，第 24 和 25 行将用户名和第一个密码显示了出来。

直接访问该页面，得到如图 6-6 所示的结果。



图 6-6　客户端未提供输入参数

修改地址栏，添加参数信息，得到如图 6-7 所示的结果。



图 6-7　读取来自客户端的信息

若结合输入页面 reg.jsp 进行测试，在表单中输入数据，然后提交到 chkReg.jsp，将得到和图 6-7 同样的结果。

6.2.5　重点提示

- 使用 request 的 getParameter 方法可以提取来自客户端的请求参数，使用时应提供参数的名称。
- request 的 getParameter 方法的返回值为 String 类型，实际应用时可能需要将此字符串转换为其他类型。
- 若提交的参数值可能为中文，则应在提取参数前使用 request 对象的 setCharacterEncoding 方法设置请求参数的编码字符集。
- 使用 out 对象可以向客户端输出信息，凡是浏览器能解释的内容，如 HTML 标签、文本、JavaScript 代码等，都可成为 out 对象的输出内容。

6.2.6　拓展与提高

1．读取一个参数的多个值

有时，从客户端传送给服务器页面的一个参数中可能包含有多个数值，通常可能是在表单中采用了复选框或列表框来收集数据的时候。例如，网页中的“新闻订阅”项，用户可能选择多个项目。为了提取一个参数的所有取值，可采用 request 对象的 getParameterValues 方法，该方法返回一个字符串类型的数组，格式为：

```
String [ ] getParameterValues(String　parameterName)
```

【例 6.2】　在页面 choice.jsp 中输入用户的订阅选项，在 printChoice.jsp 中输出用户的所有选择项。

主要代码　文件 web\6\6_2\choice.jsp

```
1：<form action="printChoice.jsp" method="post">
2：您要订阅：<br/>
3：<input name="order" type="checkbox" value="finance" />财经
4：<input name="order" type="checkbox" value="sports" />体育
5：<input name="order" type="checkbox" value="entertm"/>娱乐
6：<input name="order" type="checkbox" value="news" />时事<br/>
7：<input type="submit" value="提交" />
8：</form>
```

主要代码　文件 web\6\6_2\printChoice.jsp

```
1：<%
2：request.setCharacterEncoding("utf-8");
3：String [] ch=request.getParameterValues("order");
4：if (ch!=null)
5：{
6：    out.print("您的选择有：");
7：    for (int i=0;i<ch.length;i++)
8：    {
9：      out.print(ch[i]+" ");
10：    }
11：}
12：else
13：    out.print("请选择您要订阅的内容.");
14：%>
```

代码说明如下。

- 页面 choice.jsp 中，第 3～6 行定义了多个复选框，这些复选框的名称均为 order。
- 在接收页面 printChoice.jsp 中，第 2 行设定请求编码采用的字符集；第 3 行提取参数 order 的值，存入字符串数组 ch 中；第 4 行判断数组 ch 是否非空，若非空，则执行第 7～10 行代码，即使用 for 循环遍历数组 ch 并输出每个元素的内容，否则仅输出必要的提示。

运行结果如图 6-8 和图 6-9 所示。



图 6-8　输入多选项



图 6-9　显示用户填写的多选结果

从运行结果可以看出，复选框所取的值是由 input 标签的 value 属性所指定的。

2．读取非字符串类型的请求参数

在实际应用中，来自客户端的参数不一定是字符串类型的，而利用 request 的 getParameter 方法得到的值均为 String 类型。此时应将取得的数据转换为数值型，常用方法如下所示：

```
String sn="88",sp="29.8";
int n=Integer.parseInt(sn);
float p=Float.parseFloat(sp);
double d=double.parseDouble(sp);
```

【例 6.3】　提取数值型的参数。

在页面 inputSale.jsp 中输入本月的销售额，提交给页面 bonus.jsp，在此页面按销售额的 1%计算奖金，并显示在页面中。

主要代码　文件 web\6\6_2\inputSale.jsp

```
1：<body>
2：    <form action="bonus.jsp" method="post">
3：    本月的销售额是：
4：      <input type="text" name="sale"/>元
5：      <input type="submit" value="查看奖金数" />
6：    </form>
7：</body>
```

主要代码　文件 web\6\6_2\bonus.jsp

```
1：<%
2：      String n;
3：      n=request.getParameter("sale");
4：      float bonus=Float.parseFloat(n)*0.01f;
5：      out.print(本月奖金是："+bonus+" 元");
6：%>
```

代码分析如下。

- inputSale.jsp 中，第 2 行的<form>标签的 action 属性表示数据将提交给 bonus.jsp，第 4 行将销售额输入框的名称设为 sale。
- bonus.jsp 中，第 3 行提取名称为 sale 的请求参数的值，并赋给字符串变量 n。
- bonus.jsp 的第 4 行将 n 转换为 float 类型的数据，并按销售额的 1%的比例计算奖金数，第 5 行则输出奖金金额数。

运行结果如图 6-10 和图 6-11 所示。

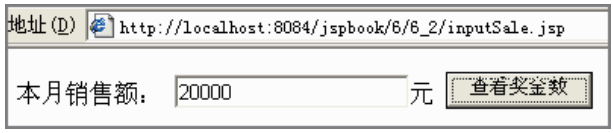


图 6-10　输入销售额

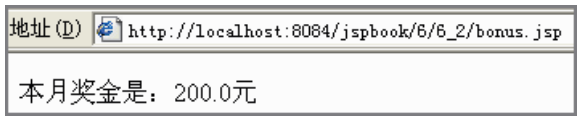


图 6-11　输出奖金数

这里仅给出了输入元素为单行文本框的情形，对于其他形式的表单元素，如密码框、单选按钮、下拉框、多行文本框、文件选择框、隐藏域等，在接收页面中获取这些元素的输入值的方法也是相同的，请大家结合前面章节的注册页面自行测试。

3．实验与思考

接下来，请大家做个小实验。直接运行 bonus.jsp，然后在地址栏的网址后面添加“?sale=1500”，这时会出现什么运行结果呢？如果添加的是“?sale=aaa”呢？

实际上，在浏览器地址栏中输入的以“?”引领的字符串相当于以 get 方式提交请求参数。以 get 方式提交数据的形式是多样的，如以下几种类型。

- 表单标签的 method 属性设为“get”，或直接省略不写，如“<form action="next.jsp" method="get">”或“<form action="next.jsp">”。
- 超链接中以“?”的形式带上参数，如“第 2 页”。
- 在地址栏中直接输入参数，如输入“http://localhost:8080/list.jsp?page=2”。

无论客户端是以 get 方式还是 post 方式提交请求参数，都可以在接收页面通过 request 对象的 getParameter 方法或 getParameterValues 方法获取参数的值。

6.3　验证用户名的唯一性（JDBC-ODBC 驱动）

在注册功能的处理中，当获取了用户填写的注册信息后，接下来就需要检测用户名的唯一性了。要验证用户名是否重复，就必须查询用户信息表，看看其中是否存在与用户新输入的用户名相同的老用户的记录。本节将学习查询数据库的一般方法。

一般地，Java 程序可使用 JDBC 访问数据库。JDBC（Java Database Connectivity）是一组可用来执行 SQL 语句的 Java API，是由 Sun 公司为 Java 制定的一系列访问数据库的类和标准接口。开发人员只需要使用这些类和接口就可以访问不同厂商出品的数据库，而不必理会这些数据库产品之间的差别。

微软公司推出的 Access 数据库功能相对简单，但使用起来方便小巧，因此这里将首先介绍访问 Access 数据库的方法。以此为引子，在熟悉了访问 Access 数据库的方法后，也就不难掌握访问其他数据库的方法了。

JDBC 中的标准接口或类均在包 java.sql 中定义，在使用这些 API 前应先导入此包。

6.3.1　使用 page 指令导入类包

使用 page 指令的 import 属性，可以在 JSP 页面中导入所需要的包或类，语法格式如下：

```
<%@ page import="包名.类名|包名.*" %>
```

上面格式中的符号“|”表示“或者”的意思。访问数据库时所需的 JDBC 接口或类均定义在包 java.sql 中，因此需要使用 JDBC 完成数据库操作的页面均应包含下面的指令：

```
<%@ page import="java.sql.*" %>
```

若一个页面中需要导入多个不同的包或类，可以使用逗号分隔，如下所示：

```
<%@ page import="java.sql.*,java.util.Date" %>
```

上面这条指令不仅导入了 java.sql 包中的所有类和接口，还导入了 java.util.Date 类。

6.3.2　查询数据库的一般过程

1．注册驱动程序

首先需要向驱动程序管理器注册数据库驱动程序。由于 Access 数据库只能使用 ODBC 驱动直接访问，因此 Sun 公司提供了从 JDBC 到 ODBC 的桥驱动，将 Java 或 JSP 程序使用 JDBC 向 Access 发起的访问转换为 ODBC 的访问方式。JDBC-ODBC 桥接驱动器作为 Java 应用的一种常用驱动程序，跟随 JDK 一起安装，完整的类名为“sun.jdbc.odbc.JdbcOdbcDriver”，因此加载 JDBC-ODBC 桥驱动的代码如下：

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

2．创建数据库连接

通过 JDBC API 访问数据库，实际上是通过向数据库管理系统（DBMS）发送 SQL 语句完成的，JSP 程序本身并不直接与数据库打交道。因此，应首先与数据库服务器建立起通信连接。一般地，可使用 DriverManager 类的静态方法 getConnection 创建到特定数据库的连接。

访问 Access 数据库时，建立数据库连接的代码如下所示：

```
Connection con = DriverManager.getConnection("jdbc:odbc:bbs");
```

这种方式要求先在 Windows 的“控制面板”→“管理工具”中创建系统数据源，上面代码中的 bbs 为数据源的名称。此时需要一些额外的手工操作，若想省去这些环节，也可改用下面的方式：

```
Connection con = DriverManager.getConnection("jdbc:odbc:driver={Microsoft Access Driver (*.mdb)};DBQ="+application.getRealPath("\")+"db\\bbs.mdb");
```

采用后面这种方式时，不需创建系统数据源，只需在代码中直接指出 Access 数据库文件的绝对路径即可。此处的 application.getRealPath("\")代表网站发布后的根路径，因此在 NetBeans 中，用到的数据库文件 bbs.mdb 应存放在项目的 web\db 子目录中。使用 MyEclipse 开发时，该数据库文件的存放路径为项目的 WebRoot\db。

3．执行查询语句（Statement 接口）

利用 Statement 接口或其子接口 PreparedStatement、CallabStatement，可以采用不同的形式

执行 SQL 语句或存储过程。此处先介绍 Statement 接口的使用方法。

1）创建 Statement 对象

利用 Connection 接口的 createStatement 方法可创建 Statement 的实例，该方法有以下两种重载方式。

方式一：

```
Statement st=con.createStatement();
```

方式二：

```
Statement st=con.createStatement(游标类型,上锁类型);
```

第二种方式中，createStatement 方法带有两个参数，分别表示游标类型与上锁类型，这两个参数的含义可参考 6.3.6 节中的相关介绍。一般情况下，使用第一种方式即可。

2）执行 select 语句

Statement 接口的 executeQuery 方法用于执行 SQL 语句中的 select 语句。执行完毕后，返回一个 ResultSet 类型的对象，该对象封装了从数据库服务器返回的查询结果。如下所示：

```
ResultSet rs=st.executeQuery("select * from book");
```

4．处理查询结果集

通过 Statement 接口执行 select 语句后，会得到 ResultSet 类型的结果集对象，它封装了从数据库返回的符合查询条件的所有行。若数据库服务器由于某种原因未能成功执行指定的 select 语句，则返回的 ResultSet 对象为 null 值。

返回的结果集有一个游标，初始状态时，这个游标定位在第一行之前。游标的移动方式由创建 Statement 对象时的参数决定，默认为 forward 方式，即只能从结果集的第 1 条记录向最后一条移动。

ResultSet 接口的常用方法有以下两种。

1）向后移动游标——next 方法

使游标指向下一行，若移动成功，则使下一行成为当前行，并返回 true 值；当游标位于整个结果集的最后一行时，执行 next 方法将返回 false 值。

由于初始状态时，游标是位于第一行之前的，故此时使用 next 方法将使游标指向结果集的第一行。

2）读取当前记录的值——getXXX 方法

对于游标所指向的当前行，可使用 ResultSet 接口的 getXXX 方法获取各列的数据，方法名称中的“XXX”表示该列的数据类型，该方法的参数为列的序号或列名。

例如，book 表的第 6 列 bprice 为 float 类型，若要显示该表第一条记录的第 6 列的值，可编写如下代码实现：

```
String sql="select * from book";
ResuleSet rs=st.executeQuery(sql);
if (rs!=null)
        if (rs.next())
            out.print("第 1 条记录的第 6 列的值是:"+rs.getFloat("bprice"));
```

这里假设已经创建了数据库的连接且已创建了 Statement 对象。上面的代码

rs.getFloat("bprice")也可改为 rs.getFloat(6)。

读取一行中各列的值时，最好按照从左到右的顺序进行。

5．关闭资源

前面所创建的 Connection 连接对象、Statement 对象、ResultSet 结果集对象，都需要占用一定的系统资源，并且相互间有一定的依赖关系。因此处理完毕后，应该按照与创建时相反的顺序关闭这些资源。可通过调用各个对象的 close 方法来关闭。

例：

```
try{if (rs!=null)
      rs.close();}catch(Exception e){}
try{if (st!=null)
      st.close();}
catch(Exception e){}
try{
      if (con!=null) con.close();
}
catch(Exception e){...}
```

6.3.3　实验案例

【例 6.4】　显示学生成绩。

这里以显示学生成绩表的数据为例，演示如何访问 Access 数据库。该数据库文件名为 student.mdb，位于项目中 Web 页的 db 子目录下，相当于发布后网站根目录下的 db 子目录，所查询的表的结构如图 6-12 所示。

score : 表

	字段名称	数据类型	
🔑	num	数字	学号，整型
	name	文本	姓名，最大长度为20的字符串
	score	数字	数值型，浮点数

图 6-12　学生成绩表的结构

程序清单　web\6\6_3\dispScore.jsp

```
1：<%@page contentType="text/html" pageEncoding="UTF-8"
     import="java.sql.*"%>
2：<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
3：       "http://www.w3.org/TR/html4/loose.dtd">
4：
5：<html>
6：  <head>
7：       <meta http-equiv="Content-Type" content="text/html;
           charset=UTF-8" />
8：       <title>数据库操作示例---成绩查询</title>
9：  </head>
10：  <body>
```

```
<%
  int num;   //用来保存学号的变量
  String name; //用来保存姓名的变量
  float score;   //用来保存成绩的变量
  Connection con = null;//声明连接对象
  Statement st = null;   //声明语句对象
  ResultSet rs = null;   //声明结果集对象
  try {
       String url;   //连接字符串
       url = "jdbc:odbc:driver={Microsoft Access Driver(*.mdb)};
        DBQ=" + application.getRealPath("/") + "db\\student.mdb";
       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
       con = DriverManager.getConnection(url); //获取连接
       st = con.createStatement();//创建 Statement 对象
       String sql = "select * from score";
       rs = st.executeQuery(sql);     //执行查询语句并得到结果
       if (rs != null) {
          out.print("<table width=300 border=1>");
          out.print("<tr><td>学号</td><td>姓名</td>");
          out.print("<td>成绩</td></tr>");
         while (rs.next()) //将游标移向下一行
         {
              //取出当前行各列的值，存储到对应的变量中
              num = rs.getInt("num");
              name = rs.getString("name");
              score = rs.getFloat("score");
              //显示记录内容
              out.print("<tr><td>" + num + "</td>");
              out.print("<td>" + name + "</td>");
              out.print("<td>" + score + "</td></tr>");
         }
         out.print("</table>");
       }
  } catch (Exception e) {
       out.print("发生数据库异常.");
       e.printStackTrace(System.err);
  } finally {   //与打开时的顺序相反，逆序关闭各项资源
       try {
            if (rs != null) {rs.close();}
       } catch (Exception e) {e.printStackTrace(System.err);}
      try{ if (st!=null) st.close();} catch (Exception e) {}
      try{ if (con!=null) con.close();}catch (Exception e) {}
  }
%>
</body>
</html>
```

代码说明如下。

- 第 1 行使用 page 指令导入了 java.sql 包。
- 第 12～14 行定义了 3 个变量，分别用以保存一条记录的 3 列的值。
- 第 15～17 行定义了连接对象、语句对象及结果集对象。
- 第 18～52 行通过 try 语句的形式完成了对表 score 的查询。第 19、20 行指出了数据库的类型及所要连接的 Access 数据库文件的绝对路径。
- 第 21 行加载了驱动类，第 22 行创建了与数据库的连接，第 23 行创建了 Statement 对象。
- 第 24 行构造了所要执行的查询语句，第 25 行执行了该语句，并将查询结果封装在对象 rs 中。
- 第 26 行判断结果集对象是否为空对象，若非空，则在第 27～29 行输出表格的表头，即标题行，并通过第 30～40 行的代码段遍历该结果集并输出每条记录的内容。
- 第 30 行代码将结果集游标移向下一行，若移动成功，则表明确实存在下一行记录。
- 第 33～35 行读取了游标所指向的当前行的 3 列的值，请注意方法名的格式。
- 第 37～39 行将取出的 3 列数据显示在表格的一行中。
- 第 43～45 行表示当发生了异常时，向客户端显示“发生数据库异常”的提示信息，并将异常信息在服务器上输出（或写到服务器的日志中）。
- 第 47～52 行关闭了打开的各个数据库资源对象，顺序与创建时相反。

访问此页面，却出乎意料地出现了如图 6-13 所示的结果。



图 6-13　显示学生成绩表时发生异常

为什么会发生异常呢？切换到工具 NetBeans IDE 中，查看代码编辑窗口下方的输出窗口，选择服务器“Apache Tomcat 6.0.26”，可以查看到运行时的控制台信息，提示信息如图 6-14 所示。

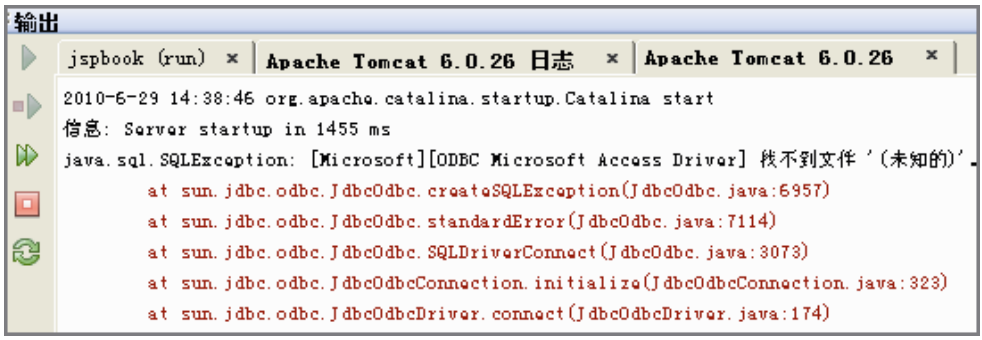


图 6-14　NetBeans 输出窗口的服务器控制台信息

输出窗口中提示“找不到文件'(未知的)'”这样的异常信息，似乎是数据库文件的路径有问题。打开 NetBeans IDE 工具中左侧的项目窗口，发现在该项目的 Web 页的 db 目录下并没有我

们要访问的 student.mdb。错误的原因找到了，修正后得到如图 6-15 所示的运行结果。

当程序执行发生异常时，查看控制台中的输出信息，往往可以帮助我们判断发生异常的原因。初学者应养成查看控制台信息的习惯，以提高自己调试程序的能力。



学号	姓名	成绩
1	程青	83.5
2	王建文	37.0
3	李观花	78.5
4	李奔	65.5
5	秦超	62.0
6	李小龙	73.0
7	徐敏艺	53.0
8	孙世青	65.7
9	王小蒙	78.9
10	金明仪	39.0

图 6-15　显示成绩表中的所有记录

请读者进行如下的实验与思考。

（1）若将第 24 行的 SQL 语句修改为“select * from score where num=1”，则将得到什么查询结果？

（2）若将第 30 行代码注释掉，将出现什么结果？为什么？

（3）将第 24 行的 SQL 语句修改为“select * from score1”，请大家按照上面的方法查看 NetBeans 工具中的输出窗口，阅读错误提示信息，找出出错的原因。

6.3.4　任务实现

前面的小节详细介绍了查询数据库的基本方法，下面回到论坛的注册功能中，完成用户名的检测工作。要验证用户输入的用户名是否唯一，也就是查询用户信息表中是否有同名的用户记录。

参考第 4 章的数据库表结构，在 Access 中创建数据库，文件名为 bbs.msb，存放在项目的 web\db 子目录下，其中用户信息表的名称为 users。

检查用户名唯一性的主要代码如下：

主要代码　web\6\6_3\chkReg.jsp

```
1：<%
2：  String uname=request.getParameter("uname");
3：  if (uname==null){
4：       out.println("请输入用户名.");
5：       return;
6：  }
7：  String sql = "select * from users where uname='" + uname + "'";
8：  //下面开始检测用户名是否可用
9：   Connection con = null;
```

```
10:    Statement st = null;
11:    ResultSet rs = null;
12:    try {
13:        String url;   //连接字符串
14:        url = "jdbc:odbc:driver={Microsoft Access Driver(*.mdb)};
            DBQ=" + application.getRealPath("/") + "db\\bbs.mdb";
15:         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
16:         con = DriverManager.getConnection(url); //获取连接
17:         st = con.createStatement();//创建 Statement 对象
18:         rs = st.executeQuery(sql);
19:           if (rs != null) {
20:              if (rs.next())
21:                  out.print("用户名已被使用，请改用其他名字吧.");
22:              else
23:                  out.print("恭喜，该用户名可以使用.");;
24:           }
25:    } catch (Exception e) {
26:              out.print("网站遇到问题，请稍后再试.");
27:              throw e;
28:        } finally {
29:             //逆序关闭资源，代码略
30:     }
31: %>
```

1．代码说明

- 第 2 行读取了用户填写的用户名，第 3～6 行判断用户名参数是否为空，若是，给出相应的提示，结束程序。
- 第 7 行构建了 SQL 语句，以查询表中所有与新用户名同名的记录。
- 第 14～17 行加载了驱动，建立了到数据库的连接并创建了 Statement 对象。
- 第 18 行通过 Statement 对象 st 向数据库服务器发送了 SQL 语句，并将得到的查询结果保存在结果集对象 rs 中。
- 第 20 行移动结果集游标，若移动成功，则表示结果集中有记录存在，也即有同名的用户记录存在，因此用户名不可用；否则说明用户名还未被使用，根据不同的检测结果输出相应的提示信息。

用户信息表中现有的记录如图 6-16 所示。

users : 表

	uname	upw	usex	udate
▶	sunny	123	女	2008-10-13
*				

图 6-16　用户信息表中的现有记录

让我们直接在 URL 中附上参数 uname，测试运行的结果，如图 6-17 所示。

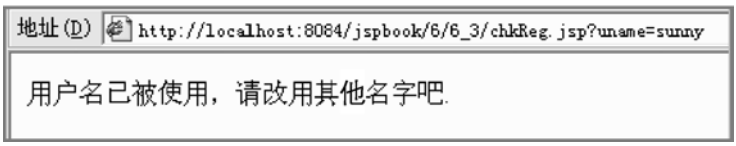


图 6-17　已经存在的用户名

让我们改用用户名 joy 进行测试，结果如图 6-18 所示。



图 6-18　测试一个无人使用的用户名

2．实验与思考

若将第 3 行的 SQL 语句改为“String sql="select * from users where uname=" + uname;”将得到什么结果？请动手实验并解释原因。

6.3.5　重点提示

1．书写正确的 SQL 语句

若发送到数据库服务器的 SQL 语句本身不正确，在执行程序前 IDE 并不会给出任何提示。因此，应保证 SQL 语句的正确性。使用断点跟踪法可以清楚地看到所执行的 SQL 语句。

2．按正确的顺序关闭资源

当查询完毕后，应按照和打开时相反的顺序去关闭数据库资源对象，但 ResultSet 对象可不必显式地关闭。

3．结果集不为 null 的意义

当结果集不为 null 时，并不能代表结果集中一定包含有记录，只表示所传送到数据库服务器的 SQL 语句已经成功执行。

4．使用 getXXX 方法获取各列的值

可使用列的序号或列名获取当前行某列的值，列号从 1 开始，从左到右编号，若使用列名获取列值，该列名不区分大小写。

6.3.6　拓展与提高

前面提到，使用 Connection 的 createStatement 方法创建 Statement 对象时，可以指定游标类型和上锁类型，这里对这两个概念做进一步的解释。

1．游标类型和上锁类型

使用 Statement 对象执行查询语句时，将得到一个符合条件的结果的集合（ResultSet 类型的对象），而创建 Statement 对象时所指定的游标类型和上锁类型都是针对这个结果集而言的。

什么是游标类型呢？得到一个结果集后，我们需要逐条处理结果集中的记录，那么哪一条才是目前正在处理的对象呢？原来，每个结果集都有自己的游标，相当于记录指针，它标识了目前正在处理的是结果集中的哪条记录。那么，这个游标能随意移动吗？当有人改变了数据库中的数据，结果集是否反映这些变化呢？遇到上述情况时，游标类型决定了游标和结果集的表现。而

上锁类型则表示该结果集是只允许读，还是可以直接通过该结果集去更新数据库。

ResultSet 中定义了一些静态常量表示这两个参数的可取值。游标类型的取值如下。

■ ResultSet.TYPE_FORWARD_ONLY（默认取值）游标只能向前滚动。

■ ResultSet.TYPE_SCROLL_INSENSITIVE 游标可随意滚动，但对他人所做的修改不敏感，即不反映数据的变化。

■ ResultSet.TYPE_SCROLL_SENSITIVE 游标可随意滚动，且对他人所做的修改敏感，即反映数据的变化。

当创建 Statement 对象时，若游标类型参数设为默认值，则使用该 Statement 对象查询得到的 ResultSet 对象只能使用 next 方法向前移动，不能使用 previous 或 absolute 等其他的方法来移动游标。

上锁类型表示结果集所允许的读写方式。它的取值如下。

■ ResultSet.CONCUR_READ_ONLY（默认取值）只读。

■ ResultSet.CONCUR_UPDATABLE 可更新，即可以使用结果集来操作数据库。

一般来说，较少直接通过结果集来改变数据库，因此通常可将上锁类型设为只读类型。

Connection 接口有一个不带参数的 createStatement 方法，此时游标类型和上锁类型取默认值，即设定关联结果集的游标类型为 ResultSet.TYPE_FORWARD_ONLY，而上锁类型为 ResultSet.CONCUR_READ_ONLY。

2．ResultSet 接口的其他方法

当指定的游标类型不是默认的 TYPE_FORWARD_ONLY 类型，如为可以滚动的类型时，结果集也提供了更多的方法来移动游标，我们也在这里做一个简单的介绍。

■ boolean previous()：将游标移向结果集的前一条记录。

■ boolean first()：将游标移向结果集的第一条记录。

■ boolean last()：将游标移向结果集的最后一条记录。

■ boolean absolute(int rows)：将游标移向指定的行号。

■ boolean relative(int rows)：以当前位置为参照，将游标移动指定的行数。

■ void beforeFirst()：将游标移到第一条记录的前面。

■ void afterLast()：将游标移到最后一条记录的后面。

■ boolean isBeforeFirst()：判断游标是否位于第一条记录的前面。

■ boolean isAfterLast()：判断游标是否位于最后一条记录的后面。

■ boolean isFirst()：判断游标是否位于第一条记录。

■ boolean isLast()：判断游标是否位于最后一条记录。

上述这些方法中，凡是返回值为 boolean 类型的，均表示游标的移动是否成功。

6.4　将注册信息插入到用户信息表中（一）

前面已经学习了查询数据库的一般方法，若需要执行数据的增、删、改操作，其基本过程与查询数据库时相同，只是所执行的 SQL 语句不同，此时执行的是 insert、delete 和 update 语句。因此不能再通过 Statement 接口的 executeQuery 方法执行了，而是要改为调用该接口的 executeUpdate 方法来执行。

假设已经创建了与数据库的连接，并创建了 Statement 实现类的对象 st，则执行记录增、

删、改的 Java 代码如下所示：

```
String sql1="insert into grade('071101101','毛毛',80)";
String sql2="delete from grade where num='0711010101' ";
String sql3="update grade set grade=grade+5 where num like '0711%'"
int n1=st.executeUpdate(sql1);
int n2=st.executeUpdate(sql2);
int n3=st.executeUpdate(sql3);
```

executeUpdate 方法的返回值是一个整数，表示执行数据更新操作后受影响的记录的行数。当用于执行 create table 或 drop table 这些不操作行的 SQL 语句时，返回的值为 0。

6.4.1　任务实现

通过上面的学习，我们可以尝试将用户的注册信息添加到用户信息表中了。结合前面几个小节的知识，得到下面的代码：

程序清单　web\6\6_4\chkReg.jsp

```
<%
 String uname,upw,upw2,usex,uemail,ulogo,msg="";
 request.setCharacterEncoding("utf-8");
 uname=request.getParameter("uname");
 upw=request.getParameter("upw");
 upw2=request.getParameter("upw2");
 usex=request.getParameter("usex");
 uemail=request.getParameter("uemail");
 ulogo=request.getParameter("ulogo");
 //先对注册信息进行基本的验证
 if (uname==null||uname.equals("")||upw==null||upw.equals(""))
  {
      msg="请填写用户名和密码.";
  }
  else if (!upw.equals(upw2))
        msg="两次输入的密码不一致!";
 if (msg.equals("")){
  Connection con = null;//声明连接对象
  Statement st= null;   //声明预编译的语句对象
  ResultSet rs=null;    // 声明结果集对象
  try {
      String url,sql;   //连接字符串
      url = "jdbc:odbc:driver={Microsoft Access Driver (*.mdb)};DBQ=" +
          application.getRealPath("/") + "db\\bbs.mdb";
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");//加载驱动程序
      con = DriverManager.getConnection(url); //获取连接
      st=con.createStatement();
      //验证用户名是否可用
      sql= "select * from users where uname='" + uname + "'";
```

```
29:          rs = st.executeQuery(sql);
30:          if (rs != null)
31:                if (rs.next())
32:                   msg="用户名<b><font color='red'> "+uname+" </font></b>
                          已被使用，请改用其他名字.";
33:          if (msg.equals("")) {//用户名可用，执行插入操作
34:                   sql="insert into users(uname,upw,usex,uemail,ulogo)
             values('"+uname+"'";
35:              sql+=",'"+upw+"','"+usex +"','"+uemail+"','"+ulogo+"')";
36:               int t=st.executeUpdate(sql);
37:     if (t>0)
38:        msg="恭喜！注册成功!";
39:     else
40:        msg="Sorry,注册失败.";
41:      }//end of 用户名可用
42:    } catch (Exception e) {
43:        out.println(""Sorry,网站出错啦."");
44:       e.printStackTrace(System.err);
45:    } finally {
46:        try{if (rs!=null) rs.close();}
47:            catch(Exception e){}
48:        try{if (st!=null) st.close();}
49:            catch(Exception e){}
50:        try{if (con!=null) con.close();}
51:            catch(Exception e){}
52:        }
53:    }
54:    //输出运行结果
55:    out.println(msg);
56:    %>
```

代码说明如下。

- 第 2 行定义了几个变量，用以存储来自表单的数据；变量 msg 用来保存处理过程中产生的提示信息。
- 第 3～9 行提取了用户填写的表单数据。
- 第 11～16 行检测必填项是否填写，两次输入的密码是否一致，并根据判断结果将提示信息存储到变量 msg 中。
- 第 17 行判断是否通过基本的验证，若正确，则通过第 18～32 行的代码判断用户名的可用性。
- 当用户名可用时，第 33～41 行代码完成了记录的插入操作。
- 第 55 行输出表示处理结果的提示信息。

将 5.2.3 节得到的注册信息输入页面 reg.jsp 文件和本小节的 chkReg.jsp 文件复制到 web 目录下，访问 reg.jsp 页面，得到如图 6-19 和图 6-20 所示的运行结果。



图 6-19　注册信息填写



图 6-20　注册成功

而当我们再次输入同一个用户名进行注册时，将得到如图 6-21 所示的结果。

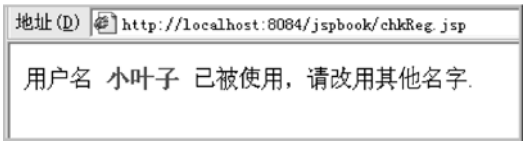


图 6-21　使用相同的用户名时，提示用户名重复

这表明在前一次操作中，用户“小叶子”的相关信息已经成功添加到了用户信息表中。

6.4.2　重点提示

executeQuery 方法与 executeUpdate 方法的使用场合。

当使用 Statement 接口执行 SQL 语句时，切记根据 SQL 语句的类型来决定使用哪个方法。若执行的是查询语句（select 语句），则使用 executeQuery 方法；若执行的是数据更新或数据定义语句（如 update 语句、insert 语句、delete 语句、create table 语句或 drop table 语句），则应使用 executeUpdate 方法。

6.5　将注册信息插入到用户信息表中（二）

6.5.1　PreparedStatement 接口

PreparedStatement 接口是 Statement 接口的子接口，称为预编译的 Statement，常用于重复

执行相似的 SQL 语句，有利于提高执行的效率。本小节改用 PreparedStatement 接口来实现用户注册功能。

下面以向图书信息表中插入一条记录为例，说明 PreparedStatement 接口的使用方法。该图书信息表有 3 列，依次为图书编号、书名和单价，分别为整型、字符串类型和浮点型。图书编号由数据库系统自动生成，在插入记录时不提供该列的值。

1．加载驱动，创建数据库连接

```
String url = "jdbc:odbc:driver={Microsoft Access Driver (*.mdb)};DBQ=" + filepath;
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");//加载驱动程序
    con = DriverManager.getConnection(url); //获取连接
```

这里的 filepath 是 Access 数据库文件的绝对路径。

2．构建 SQL 语句

需要传送到数据库服务器的 SQL 语句，其中的 IN 参数值先用占位符?表示。

```
String sql = "insert into book(bname,bprice) values(?,?)";
```

3．创建 PreparedStatement 对象

```
PreparedStatement pt = con.prepareStatement(sql);
```

4．设置占位符的值

依次指明 SQL 语句中各个占位符的值，方法名称需根据表中该列的类型而定。

```
pt.setString(1, "人性的弱点");     //设置第 1 个?的值，表示书名
pt.setFloat(2, 26.8);              //设置第 2 个?的值，表示单价
```

5．执行 SQL 语句

executeUpdate 方法返回受影响的记录行数，而 executeQuery 方法则返回 ResultSet 对象。

```
int n=pt.executeUpdate();
```

6．逆序关闭对象

```
try{if (pt!=null) pt.close();}
catch(Exception e){}
try{if (con!=null) con.close();}
catch(Exception e){}
```

上面的 6 个步骤是使用 PreparedStatement 接口执行 SQL 语句的一般过程，该接口的 setXXX 方法用于设定参数的值，其原型如下：

```
void setXXX(int index,Object o)
```

set 方法名称中的 XXX 由参数的类型决定。例如，本例中的第 2 个占位符，表示的是图书单价，建表时设计为 float 类型。因此，设置这个参数的方法名为 setFloat。

该方法的第 1 个参数为 SQL 语句中出现的占位符的序号，第 1 个占位符的序号为 1，第 2 个为 2，以此类推。

该方法的第 2 个参数表示赋给该占位符参数的值。

PreparedStatement 也可用来执行 delete、update、select 等其他 SQL 语句。执行 select 语句

时应使用 executeQuery 方法，仍返回一个 ResultSet 类型的结果集。

6.5.2　实验案例

【例 6.5】　添加学生成绩记录。

本案例使用 PreparedStatement 接口执行 SQL 语句，向学生成绩表中添加一条记录。代码如下：

程序清单 web\6\6_5\insertScore2.jsp

```
1：<%@page contentType="text/html" pageEncoding="UTF-8"
        import="java.sql.*"%>
2：//此处略去部分代码
3：<body>
4：  <%
5：      Connection con = null;//声明连接对象
6：      PreparedStatement pt= null;   //声明预编译的语句对象
7：      /*---变量 filepath: 表示数据库文件的绝对路径----*/
8：    String filepath= application.getRealPath("/") + "db\\student.mdb";
9：    try {
10：        String url;   //连接字符串
11：        url = "jdbc:odbc:driver={Microsoft Access Driver(*.mdb)};
                DBQ=" +filepath;
12：        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
13：        con = DriverManager.getConnection(url);
14：        //将要执行的 SQL 语句,具体数据先用占位符?表示
15：        String sql="insert into score values(?,?,?)";
16：        //创建预编译语句对象并将 SQL 语句载入其中，准备执行
17：        pt=con.prepareStatement(sql);
18：        pt.setInt(1, 30);   //设置 SQL 语句中第 1 个?的值，此处为学号的值
19：        pt.setString(2, "黄小敏"); //设置 SQL 语句中第 2 个?的值
20：        pt.setFloat(3, 90); //设置 SQL 语句中第 3 个?的值，此处为成绩
21：        //执行 SQL 语句，并将执行成功的记录条数赋值给变量 t
22：        int t=pt.executeUpdate();
23：         if (t>0)
24：            out.println("数据添加成功!");
25：         else
26：              out.println("Sorry,数据添加失败.");
27：          } catch (Exception e) {
28：                    out.println("添加数据时发生异常.");
29：                    e.printStackTrace(System.err);
30：          } finally {   //与打开时的顺序相反，逆序关闭各项资源
31：                try{if (pt!=null) pt.close();}
32：                catch(Exception e){}
33：                try{if (con!=null) con.close();}
34：                catch(Exception e){}
35：          }
36：           %>
```

```
37：　　</body>
38：</html>
```

代码中已有较详细的注释，这里不再重复说明。

6.5.3　任务实现

按照下面的提示，修改 6.4.1 节中的 chkReg.jsp 页面的代码。

（1）在代码“Connection con = null;”后面添加下面的代码：

```
PreparedStatement pt=null;
```

（2）在 chkReg.jsp 中找到下面的代码段（第 34～36 行）：

```
sql="insert into users(uname,upw,usex,uemail,ulogo)
        values('"+uname+"'";
sql+=",'"+upw+"','"+usex +"','"+uemail+"','"+ulogo+"')";
        int t=st.executeUpdate(sql);
```

用下面的代码段替代：

```
sql="insert into users(uname,upw,usex,uemail,ulogo) values(?,?,?,?,?)";
pt = con.prepareStatement(sql); //创建预编译对象，载入 SQL 语句
pt.setString(1, uname); //设置上面第 1 个?的值，即用户名
pt.setString(2, upw);    //设置第 2 个?的值，即密码
pt.setString(3, usex);   //性别
pt.setString(4, uemail); //E-mail 地址，注册时间直接由数据库中取默认值
pt.setString(5, ulogo);   //用户头像文件的路径
int t=pt.executeUpdate(); //执行 SQL 语句，即把数据插入到表中
```

（3）在关闭 con 对象之前添加关闭对象 pt 的代码，如下：

```
try{if (pt!=null) pt.close();}
catch(Exception e){}
```

重新测试注册页面，将得到与 6.4 节一样的结果。

6.5.4　重点提示

- 使用 PreparedStatement 对象执行 SQL 语句时，其中的参数可先用占位符?表示，但在执行前，应为所有的占位符设置具体的值。
- 在使用 setXXX 方法设置参数值时，方法名应根据参数的类型而变化。setXXX 方法的第 1 个参数表示在 SQL 语句中出现的占位符？的序号，第 1 个参数的序号为 1。
- 接口名称为 PreparedStatement，而创建该类型的对象的方法名称为 prepareStatement，请注意二者拼写上的差别。

6.6　页面跳转控制

用户注册成功或失败后，在给出相应的提示信息后，应跳转到合适的页面，以便引导用户的下一步操作。在 JSP 技术中，可以通过转发请求或重定向网页达到控制页面转向的目的。

6.6.1　<jsp:forward>动作

我们可以使用<jsp:forward>动作实现请求的转发，那么什么是 JSP 的动作呢？

JSP 的标准动作都是形如<jsp: XXX >这样的标记，它们遵循 XML 的语法规范。一个标准动作代表了 JSP 规范中已经标准化的某种操作。

在 JSP 引擎将页面转换成 Servlet 时，若遇到了一个 JSP 的标准动作，则将调用一段标准的 Java 代码。因此，JSP 的标准动作在处理请求时起作用，影响到 JSP 运行时的行为和发送给客户端的应答。JSP2.0 标准中约有 20 项标准动作，我们将陆续学习最常用的几个动作。

<jsp:forward>动作直接在服务器端将用户的请求转发到另一个页面处理，同时来自客户端的请求参数等数据也一并被转发，该动作发生时并不通知客户端，客户端对此一无所知。

其语法格式为：

```
<jsp:forward page="relativeURL"/>
```

或

```
<jsp:forward page="relativeURL">
    <jsp:param name="name" value="value" />
</jsp:forward>
```

page 属性指明了要将请求转给哪个页面（或其他资源）处理，采用第 2 种方式时，还可以向转发过去的页面传送一些新的参数。

【例 6.6】　请求转发示例。

当客户端向 index.jsp 页面发出请求时，此页面将请求转给 login1.jsp 进行处理，同时传递一个新的参数，该参数的名称为 utype，值为 admin。

主要代码　文件 web\6\6_6\index.jsp

```
1：<body>
2：      <jsp:forward page="login1.jsp">
3：           <jsp:param name="utype" value="admin"/>
4：      </jsp:forward>
5：      <br/><br/><br/>
6：      主页面内容区域
7：</body>
```

主要代码　文件 web\6\6_6\login1.jsp

```
1：<body>
2：      <%
3：           String usertype = request.getParameter("utype");
4：           if (usertype.equals("admin")) {
5：                out.println("管理员登录模拟页面.");
6：            } else {
7：                out.println("普通用户登录模拟页面.");
8：           }
9：     %>
10：</body>
```

访问页面 index.jsp，得到如图 6-22 所示的结果。



图 6-22　根据参数值进行请求转发

观察运行结果，你是否留意到了呢？

虽然浏览器窗口显示的是 login1.jsp 页面的内容，但是地址栏显示的地址仍是原来的 index.jsp。这表明向 login1.jsp 发起的请求并未经过客户端，而是由服务器直接将客户端的请求转发给 login1.jsp 处理的。当然，也是由 login1.jsp 负责向客户端发回响应的。

页面 index.jsp 中在<jsp:forward>动作之后的内容未被显示，因为请求已经被转发给 login1.jsp 处理，因此在<jsp:forward>之后的代码不会被执行。

此外，若不需向转发的页面传递参数，则应采用<jsp:forward />的形式，如:

```
<jsp:forward page="next.jsp" />
```

既然客户端的请求被转发到另一个页面，那么客户端提交给原页面的请求参数等信息是否也一并被转发了呢？实践是最好的证明，下面就通过实验来证明这一点。

修改 login1.jsp，添加代码段，以便提取和显示请求参数 msg 的值，修改后的代码如下：

```
1：<body>
2：    <%
3：        String usertype = request.getParameter("utype");
4：        String msg;
5：        msg=request.getParameter("msg");
6：        if (msg==null) msg="";
7：        if (usertype.equals("admin")) {
8：            out.println("管理员登录模拟页面.附加信息："+msg);
9：        } else {
10：            out.println("普通用户登录模拟页面.附加信息："+msg);
11：        }
12：    %>
13：</body>
```

将该页面保存为 login2.jsp，修改原 index.jsp 中的<jsp:forward />动作，将其 page 属性改为指向 login2.jsp。

在地址栏中输入下面的网址进行测试，结果如图 6-23 所示。

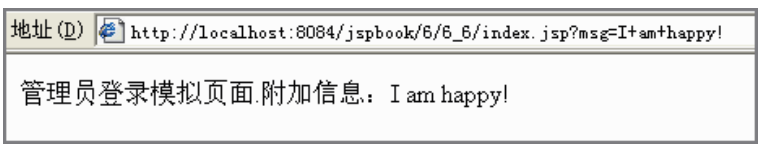


图 6-23　转发请求的同时，参数也被转发

从上图的运行结果可以看出，当请求被转发给 login2.jsp 时，我们传送给 index.jsp 的参数

msg 也被传送过去了。此时，从另一角度来看，对于客户端发起的这一次请求，服务器端需执行 2 个页面来响应，因此也可以称 login2.jsp 和 index.jsp 是同一次请求范围内的页面，它们所使用的 request 对象是相同的。

6.6.2　response 对象的重定向方法

response（响应）对象是 JSP 的内建对象之一，封装了从服务器返回给客户端的响应信息，包括 HTTP 状态行、响应标头和消息体。

使用 response 的 sendRedirect 方法或 setHeader 方法可实现网页的重定向。

1．void sendRedirect(String location)

该方法可将客户端的请求重新定向到另一个位置去。它实际上是向浏览器发送一个控制信息，指示浏览器向由 location 参数指定的网页发出新的请求。因此，上一次请求中的信息都会丢失。

例如，重定向到 login.jsp：

```
<%
    response.sendRedirect("login.jsp");
%>
```

例如，重定向到 Google 网站：

```
<%
    response.sendRedirect("http://www.google.com");
%>
```

需要留意的是，执行了 sendRedirect 方法后，页面的其他代码可能仍会被执行。因此，若需要终止剩余代码的执行，可在该语句之后跟随一个 return 语句。

2．void setHeader(String name,String value)

可通过设置 refresh 响应标头向客户端发送一个重定向信息，使浏览器在指定时间之后自动重定向到其他网页资源。

例如，显示提示信息，并在 3 秒后重定向到 login.jsp 页面：

```
<%
    out.print("注册成功！3 秒后自动跳转到登录页面...");
    response.setHeader("refresh","3;url=login.jsp");
%>
```

从该例可以看出，refresh 标头的值应按“秒数;url=重定向网页的 URL”这种形式设置，注意秒数后面是分号。

又如，2 秒后重定向到 http://www.baidu.com，则应写成：

```
response.setHeader("refresh","2;url= http://www.baidu.com");
```

6.6.3　任务实现

前面我们学习了控制页面跳转的两种方式，接下来让我们为前面完成的注册页面添加后继的跳转控制。当注册成功时，显示成功信息并跳转到网站的首页；若注册失败，则返回注册页

面，用户名、E-mail 等输入框维持原来的输入值，并在表单上方显示错误提示信息。

此时，注册输入页面 reg.jsp 和数据处理页面 chkReg.jsp 都要做相应的修改，下面只给出有变化的代码，其余代码请参考 6.4.1 小节或 6.5.3 小节的相关页面文件。

修改后的代码　web\6\6_6\reg.jsp

```
1：<%
2：     String msg = "", uname = "", uemail = "", ulogo = "";
3：      if (request.getParameter("msg") != null) {
4：         msg = request.getParameter("msg");
5：         uname = request.getParameter("uname");
6：         uemail = request.getParameter("uemail");
7：         ulogo = request.getParameter("ulogo");
8：      }  %>
9：<body>
10：//其他代码略
11：     <div class="warning" ><%=msg%></div>
12：// ...
13：     <input type="text" name="uname" value="<%=uname%>"
           size="40" maxlength="20"   />
14：     <form>
15：//其他代码略
```

当注册失败后重新返回注册输入页面时，应在输入框中显示上一次输入的内容，因此应给输入框添加 value 属性，如代码中的第 13 行所示。

而在页面 chkReg.jsp 中，将原来的代码“out.println(msg);”用如下所示的代码段替换：

修改后的代码　web\6\6_6\chkReg.jsp

```
1：     if (msg.equals("恭喜！注册成功!")){out.println(msg);
2：          response.setHeader("refresh","3;url=login.jsp");
3：          }
4：     else  {
5：     %>
6：     <jsp:forward page="reg.jsp">
7：          <jsp:param name="msg" value="<%=msg%>" />
8：     </jsp:forward>
9：     <%}// end of else%>
```

重新测试注册页面，当我们输入前面已经注册过的用户名时，如图 6-24 所示，在表单上方显示了错误提示信息。

从运行时的情况看，似乎是输入完毕并提交后，这个提示信息就出现在同一个注册页面上了，而实际上，注册信息是提交到 chkReg.jsp 页面进行处理的，当发生“用户名重复”这种情况时，chkReg.jsp 页面又将请求重发回原来的注册输入页面，因此看起来好像从来没离开过这个页面一样。

而当注册成功时，则会自动转向登录页面。



图 6-24　通过转发在输入页面显示了错误提示

6.6.4　重点提示

无论是重定向还是用<jsp:forward>动作实现的请求转发，呈现给客户端的效果都是页面的“跳转”，但二者之间存在着不可忽视的差别。

- 将页面 A 重定向到页面 B 时，原来客户端发送给 A 页面的请求信息将丢失；而服务器将客户端对 A 页面的请求转发给 B 页面时，会将请求信息同时转发给 B。
- 请求只能转发给同一应用的其他网页，但重定向则无此限制，可以重定向到 Internet 的其他资源上，当然也可以定向到同一服务器的其他应用中。



思考与训练

一、基础训练

1．写出 page 指令，设定页面的编码为 GB2312。

2．定义字符串变量 name，提取请求参数 uname 的值，并赋给 name。

3．定义 float 类型的变量 price，提取请求参数 bprice 的值，保存到 price 中。

4．若请求中包含参数 uname，则输出“欢迎 XXX”，其中 XXX 是该参数的值，否则输出“请输入您的名字”。

5．设置请求所采用的编码字符集为 UTF-8。

6．若有代码“下一页”，则在 next.jsp 中可以提取到什么参数？请编写代码输出这些参数的值。

7．重定向到页面 list.jsp，同时发送参数，名称为 pageno，参数值由变量 pno 表示。

8. 当请求参数 lang 的值为“chn”时，将请求转发到 index_chn.jsp，否则转发到 index_en.jsp。

9．在页面上显示文字“您尚未登录.”，并在 2 秒后自动重定向到登录页面 login.jsp。

10．写出 jsp 动作。将请求转发到 list.jsp，且传递给 list.jsp 页面一个参数，名称为 type，

值为“news”。

11．连接 Access 数据库，文件路径为 e:\data\shop.mdb。

12．设有表 movie(no,name,up_date,owner)，分别表示视频的编号、片名、上传日期、上传者。现欲查询片名 name 中含有“末路”二字的所有视频的信息，请写出该查询语句，并保存到字符串变量 sql2 中。

二、思考题

1．什么是客户端的一次请求？request 对象封装了哪些信息？

2．out 和 response 对象各有什么作用？有哪些常用方法？

3．<jsp:param>动作可结合哪些动作一起使用？有何作用？请写出其语法格式。

4．<jsp:forward>动作所实现的转发与使用 response 对象的 sendRedirect 方法所实现的重定向有何异同？

三、综合实训

1. 模拟商品订购中的一个简单功能：在页面 input.jsp 中输入数量和单价，提交到 money.jsp 计算金额。在 money.jsp 中，若用户提交的不是数值型数据，则需捕捉该类型的异常（NumberFormatException），并输出提示信息“请输入正确的数值”；若无异常，则计算总金额并输出。

2．假设居民户籍信息表 resident 结构如下：

列　名	含　义	类　型
rno	户口编号	char(8)
raddr	住址	varchar(50)
rhost	户主姓名	varchar(30)
rdate	入户日期	char(10)
rnum	本户人数	int

请在 Access 中创建该表，并编写 JSP 页面，以表格的形式显示表中前 10 条户主姓“王”的记录。

3．完成论坛的发帖功能，在帖子发布成功后，跳转到该帖子的详情页面，显示刚发表的帖子的标题和内容。

第 7 章　登录功能的实现

7.1　任务的提出与分解

用户注册成功后，即可使用已注册的用户名与密码登录网站，以完成身份的验证。登录功能的完成可分解为以下几个子任务。

- 制作登录信息输入页面。
- 验证用户输入的用户名与密码是否正确。
- 用户登录成功后，在其访问本网站的一次会话期间，维持其已登录的状态。
- 显示当前在线的人员名单。

登录信息输入页面的制作可参考第 5 章，这里不再详述。

7.2　验证登录信息（JDBC 驱动）

验证登录信息是否正确，实际上就是查询用户信息表中是否存在一条记录，其用户名和密码与用户输入的信息相同。本章我们将介绍直接通过 JDBC 驱动访问数据库的方法。

7.2.1　通过 JDBC 访问数据库

前文提及，JDBC（Java Database Connectivity）是 Sun 公司提供的一套访问数据库的标准类或接口。目前，JDBC 通用接口得到了各大数据库厂商的支持，主流的数据库产品如 Oracle、Microsoft SQL Server、DB2、MySQL 等都实现了这些标准接口，并把这些实现类全部打包成 jar 文件提供给开发人员使用。我们平常所说的 JDBC 驱动程序就是指这些 jar 文件，它们实现了 JDBC 标准接口和类的集合。若项目中需要访问数据库，则应添加对这些 jar 文件的引用。如图 7-1 所示展示了使用 JDBC 访问数据库时资源之间的调用关系。

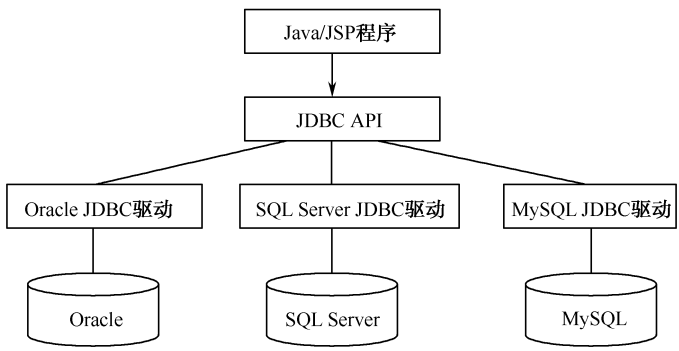


图 7-1　JDBC 访问数据库

一般来说，可以在数据库产品开发商的官方网站上下载合适的 JDBC 驱动程序。

得到 JDBC 驱动包后，首先应在项目中添加对该驱动包的引用。在 NetBeans IDE 中，可

在项目视图中选择“库”，在其右键菜单中选择“添加 JAR/文件夹”，将之前下载的 jar 驱动文件添加到项目中来，如图 7-2 所示。



图 7-2　为项目添加必需的驱动文件

若使用 MyEclipse 进行开发，则应在项目的构建路径（Build Path）中添加对相关 jar 文件的引用。

接下来，使用 JDBC 驱动访问数据库的过程与第 6 章中使用 JDBC-ODBC 桥驱动访问数据库的过程基本相同。下面给出了连接 MySQL 数据库的关键代码，这里假设所连接的数据库名称为 bbs，连接数据库服务器的用户名是 root，密码是 rootroot，这些参数的值在使用时应根据实际情况调整。

```
1：String driverClass="org.gjt.mm.mysql.Driver";
2：String url=" jdbc:mysql://localhost:3306/bbs?
   useUnicode=true&characterEncoding=utf8";
3：String username="root";
4：String password="rootroot";
5：Connection con=null;
6：try{
7：Class.forName(driverClass);
8：con=DriverManager.getConnection(url,username,password);
9：...
10：}
11：catch(SQLException e)
12：{e.printStackTrace(System.err);}
13：finally{
14：    try{ if (con!=null) con.close();}
15：    catch(Exception e){}
16：}
```

不过，当所采用的数据库产品不同时，连接数据库的驱动类全称和 URL 也不相同。下面给出了几种常用的数据库产品的驱动类名称和连接 URL。代码段中出现的“localhost”是数据库服务器的名称，设为“localhost”则表示数据库服务器与 Web 服务器安装在同一台机器上。若数据库服务器安装在其他机器上，一般可使用该机器的 IP 地址或域名。

■ 连接 Microsoft SQL Server 2000：

```
String driverClass =" com.microsoft.jdbc.sqlserver.SQLServerDriver ";
String url=" jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=数据库名称";
```

■ 连接 Microsoft SQL Server 2005：

```
String driverClass ="com.microsoft.sqlserver.jdbc.SQLServerDriver";
String url="jdbc: sqlserver://localhost:1433;DatabaseName=数据库名称";
```

■ 连接 Oracle：

```
String driverClass ="oracle.jdbc.driver.OracleDriver";
String url="jdbc:oracle:thin:@localhost:1521:实例名";
Oracle 的默认实例名一般为 orcl.
```

7.2.2　任务实现

这里以访问 MySQL 数据库为例，实现用户登录信息的验证。

首先，应安装 MySQL 服务器软件，安装过程中需注意选择所使用的默认字符集，一般应设为 utf-8。若安装时未选择，也可在安装完毕后修改 MySQL 的配置文件 my.ini 重新设置。此外，应记录安装过程中设置的连接用户名和密码。

之后，为了更方便地管理和操作数据库，可选择安装一个 MySQL 的图形界面管理工具，如 SQLYog 或 SQLManageLite 等工具软件。

最后，启动 MySQL 服务器，打开图形管理工具，创建数据库 bbs，注意将数据库所采用的编码字符集设为 utf-8，并按照第 4 章所设计的数据库结构，在库 bbs 中建立论坛网站前台子系统所需的 3 张表，即用户信息表 users、主题帖表 posts、回复帖表 reply，并在每张表中添加数条测试用的记录。

若无图形管理工具，也可以在 MySQL 安装完毕并启动后，在 MySQL 程序组中选择“MySQL Command Line Client”。启动 MySQL 命令行模式的客户端，在其中输入 SQL 语句来创建数据库和表。

数据库创建完毕后，在项目中添加对 MySQL 驱动 mysql-connector-java-3.1.12-bin.jar（不同版本时文件名称可能会有不同）的引用，然后编写页面代码。验证登录信息的页面 chkLogin.jsp 的主要代码如下：

主要代码　web\7\7_2\chkLogin.jsp

```
1：<%
2：    String uname, upw;
3：    uname = request.getParameter("uname");
4：    upw = request.getParameter("upw");
5：    if (uname == null || uname.equals("")) {
6：        out.println("请输入用户名.");
7：        response.setHeader("refresh", "3;url=login.jsp");
8：        return;
9：    }
10：    if (upw == null || upw.equals("")) {
11：        out.println("请输入密码.");
12：        response.setHeader("refresh", "3;url=login.jsp");
```

```
13:     return;
14:  }
15:  boolean valid = false;     //表示验证是否通过的标志
16:  String driverClass = "org.gjt.mm.mysql.Driver";
17:  String url = "jdbc:mysql://localhost:3306/bbs?
         useUnicode=true&characterEncoding=utf8";
18:  String username = "root";
19:  String password = "rootroot";
20:  Connection con = null;
21:  Statement st = null;
22:  ResultSet rs = null;
23:  try {
24:     String sql = "select * from users where uname='" + uname + "'";
25:     Class.forName(driverClass);
26:     con = DriverManager.getConnection(url, username, password);
27:        st = con.createStatement();
28:        rs = st.executeQuery(sql);
29:        if (rs != null) {
30:          if (rs.next()) {
31:               valid = upw.equals(rs.getString("upw"));
32:            }
33:        }
34:     } catch (SQLException e) {
35:        e.printStackTrace(System.err);
36:     } finally {
37:     //此处略去部分代码，依次关闭对象 rs、st、con.
38:     }
39:     if (valid) {
40:        out.println("登录成功!正在自动转向网站首页...");
41:        response.setHeader("refresh", "3;url=index.jsp");
42:     } else {
43:        out.println("用户名或密码错误.");
44:       response.setHeader("refresh", "3;url=login.jsp");
45:     }
46:  %>
```

代码说明如下。

- 第 5～14 行判断用户名与密码是否为空，若未提供其中的一项参数，则给出提示信息，并自动重定向到登录页面。
- 第 15 行定义了变量 valid，用来表示验证是否通过，默认值为 false，表示不通过。
- 第 16～19 行定义了访问 MySQL 数据库 bbs 所需要的连接参数。
- 第 25～28 行完成了数据库连接，并执行了查询语句，查询结果保存到结果集对象 rs 中。
- 第 29～33 行判断了结果集 rs 中是否有记录，若有，则判断该条记录的密码是否与用户所输入的密码相同，若一致，则将验证标志修改为 true。

- 第 39～45 行对验证成功或失败两种情况给出了相应的提示信息，并转向网站首页或登录页面。

7.2.3　重点提示

- 以 JDBC 方式访问数据库前，应先在项目中添加对相应的数据库驱动文件，即 jar 文件的引用，否则，将产生 java.lang.ClassNotFound 异常，表示无法加载所需要的类。
- 对于不同厂商提供的数据库产品，所需注册的驱动类名称和创建数据库连接时的 URL 也不相同。

7.3　在会话期间维持用户的登录信息

当用户通过登录页面登录成功后，用户随后所访问的其他页面都应能得知“该用户已经登录”的这个事实。例如，用户登录后，一般地，网站将会在随后的页面中显示登录用户名，同时，用户还具有发帖等权利。

在第 2 章中我们了解到，HTTP 协议是无状态的协议，不具备事务记忆的能力，通过协议本身无法保持客户端前一次请求产生的状态信息，因此后继的请求也无法获取之前的请求所产生的状态信息。在“用户登录”这个场景中，无法通过 HTTP 协议告知后面的页面“用户已经登录”这一事实，因此需要通过其他手段来实现这一目的。JSP 内置的 session（会话）对象就是专门用来保存客户端的会话状态的。

7.3.1　session 对象的工作机制

session（会话）对象是 JSP 的内建对象之一，用于在客户端与服务器的一个会话期间保持客户端的信息。所谓“会话周期”，是指从客户端连接到服务器（如访问服务器上的某一网页）开始，一直到发生以下 4 种情形之一的这段时间。

- 用户关闭了目前正在使用的浏览器程序。
- 服务器被关闭（如 Tomcat 服务被终止）。
- 在指定长度的时间段里，用户未向服务器提出请求。Tomcat 服务器默认为 30 分钟。
- 程序中遇到结束 session 的代码。

当会话开始时，服务器为客户端创建一份专属的 session，即在服务器机器上留出一块内存区域供该客户端使用，该客户端所访问的任何网页都有权写入或读取其中的信息。我们可以理解为，客户端与服务器之间有一份“会话记录”，里面保存双方这次会话期间的一些数据，该客户端无论访问应用的哪一项资源，该资源都有权查看这份会话记录。当然，也有权向其中写入数据。

为了与其他客户端的 session 区别，服务器用一个 ID（即 sessionID）来标识它。当服务器向客户端发回响应时，该 ID 将连同响应内容一起发送给客户端，客户端通常可通过 Cookie 机制保存此 ID。以后当用户访问网站的其他网页（或其他资源）时，每次都会将 sessionID 夹带在其他请求信息中一起发送给服务器。这样，无论用户访问的是哪项资源（一般是网页），服务器都可识别出专属于该客户端的 session。

若 Cookie 被禁用，则会通过 URL 重写的方式向服务器端发送 sessionID。

切记，session 保存的是单个客户端的信息，每个客户端都只能访问属于自己的那份 session。

session 对象的常用方法如下。

1．void setAttribute(String name,Object value)

向 session 中写入一个属性，该属性的名称由第 1 个参数 name 表示，值由第 2 个参数 value 表示，该值可以是一个 Java 对象。若同名的属性已经存在，则将覆盖原来的值。

例如，将请求参数 username 的值以 uname 的名称存入 session 中：

```
<%
    session.setAttribute("uname",request.getParameter("username"));
%>
```

2．Object getAttribute(String name)

返回名称为 name 的属性的值，返回值为 Object 类型，通常需根据具体应用进行强制类型转换。若不存在指定名称的属性，则返回 null，如“<% String name=(String)session.getAttribute ("username");%>”。

3．void removeAttribute(String name)

删除 session 中指定名称的属性。

例如，在用户退出登录时，从 session 中清除所保存的用户名：

```
<%
        session.removeAttribute("username");
%>
```

4．void setMaxInactiveInterval(int interval)

设置所允许的最长的非活动时间间隔，有人形象地称之为“发呆时间”，单位为秒。

为了更有效地利用服务器的资源，若客户端在规定长度的一段时间内都未访问服务器，则服务器将终止与该客户端的会话，并清除与该客户端相关联的那份 session。Tomcat 服务器默认的会话超期时间为 30 分钟，可在项目的配置文件 web.xml 中自行设置。

例如，将发呆时间设为 1 小时：

```
<%
    session.setMaxInactiveInterval(60*60);
%>
```

5．void invalidate()

销毁当前 session 对象，执行此方法后，将不能再使用 session 对象。

7.3.2　实验案例

【例 7.1】　使用 session 实现简单的点菜功能。

这里为客户提供 3 个菜单页面，分别可以选择汤类、色拉及特色菜。当用户在其中一个页面完成了点菜选择后，将自动转向下一个选择页面。每个选择页面都将完成相同的功能：将用户在前一个页面做出的选择存入 session 中，并且显示出已点的菜单，同时还提供一个菜单导航条。我们设计一个公共的页面 addToSession.jsp 来完成这些相同的功能，代码如下：

```
程序清单　文件 web\7\7_3\menu\addToSession.jsp
1：<%@page contentType="text/html" pageEncoding="UTF-8"%>
2：<%
```

```
3:      request.setCharacterEncoding("utf-8");
4:      String selemenu[] = request.getParameterValues("food");
5:      String list = (String) session.getAttribute("list");
6:      if (selemenu != null) {
7:              if (list == null) {
8:                      list = "";
9:               }
10:              for (int i = 0; i < selemenu.length; i++) {
11:                      list += selemenu[i]+";   ";
12:              }
13:              session.setAttribute("list", list);
14:         }
15:          if (list != null) {
16:                 out.println("您已经点了：" + list+"<br/>");   //显示点菜清单
17:          }
18: %>
19: <a href="salad.jsp">色拉</a> 
20: <a href="dish.jsp">特色菜</a> 
21: <a href="soup.jsp">汤类</a><br/>
```

代码分析如下。

- 第 4 行代码取出用户提交的选择，存入数组 selemenu 中。
- 第 5 行则取出 session 中的点菜单，存入字符串变量 list 中。
- 第 6 行判断用户是否有提交的数据，若无，则转去 15 行显示已点的菜单。
- 第 7 行判断 list（即点菜单）是否非空，若为空，则将其内容设为空字符串。
- 第 10～12 行取出数组 selemenu 中的各项数据，连接到字符串 list 中。
- 第 13 行将 list 重新存入 session 中，以供其他页面使用。
- 第 15～17 行当 list 非空时，输出其中内容，即已经点了的菜单清单。
- 第 19～21 行输出 3 个超链接，可链接到 3 个选择页面。

3 个菜单选择页面（汤类选择、色拉选择、特色菜选择）基本是相同的，只是具体的选择内容有所不同，若结合数据库的查询技术，可以将它们合并为一个页面程序来处理。本例重点演示 session 对象的使用方法，故菜单清单仅采用静态页面的形式呈现。

下面列出了 salad.jsp 页面的主要代码，其中第 17 行代码包含了页面 addToSession.jsp 的内容，之后是以复选框形式提供的色拉选择表单：

主要代码　文件 web\7\7_3\menu\salad.jsp

```
1: <body>
2:     <%@include file="addToSession.jsp" %>
3:     <hr color="yellow">
4:     <p><img border="0" src="jpg\salad.jpg" width="100" height="80"/></p>
5:     <form action="dish.jsp" method="post">
6:         <input type="checkbox" name="food" value="大虾沙拉"/>大虾沙拉
7:         <input type="checkbox" name="food" value="蟹肉沙拉">蟹肉沙拉
8:         <input type="checkbox" name="food" value="什锦沙拉">什锦沙拉
9:         <input type="checkbox" name="food"value="蔬菜沙拉">蔬菜沙拉<br/>
```

```
10：          <p><input type=submit value="确定"></p>
11：      </form>
12：</body>
```

访问此页面，运行效果如图 7-3 所示。



图 7-3　选择色拉的页面

当用户在每个页面点菜并提交后，将在下一个页面上看到目前所点的菜单。最后，在汤类页面提交选择后，将得到如图 7-4 所示的效果。

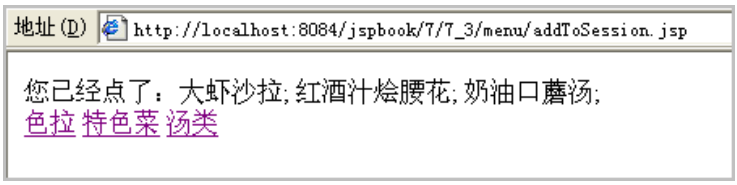


图 7-4　点菜完毕后显示的点菜单

该案例的功能类似于购物车，不过在实现购物车的功能时，一般使用 ArrayList 对象或 Vector 对象来保存用户选购的多个商品的信息。

7.3.3　任务实现

在网站的登录功能中，为了使用户在通过登录验证后，在访问其他页面时继续维持“已经登录”的这个状态信息，应在登录成功后将用户名写入 session 对象中。修改 7.2.2 节的登录验证页面 chkLogin.jsp，将登录成功后的处理代码修改为：

```
if (valid) {
        session.setAttribute("username",uname);
        out.println("登录成功!正在自动转向网站首页...");
        response.setHeader("refresh", "3;url=index.jsp");
}
```

修改后，登录成功后系统会将用户名保存到 session 对象中。

在 index.jsp 页面的<body>标签后添加如下代码：

```
<%
    if (session.getAttribute("username") == null) {
        response.sendRedirect("login.jsp");
        return;
        }
    out.print("欢迎" + session.getAttribute("username"));
    //其他代码
%>
```

重新打开浏览器窗口，访问 index.jsp，由于此时在 session 对象中不能找到名称为 username 的属性，因此会令客户端重定向到登录页面，而当登录成功后，再次重定向到 index.jsp 页面，此时可以在 session 中查到名称为 username 的属性，因此得知用户已经登录，于是在页面上显示了用户名。

用户登录后 index.jsp 页面的运行效果如图 7-5 所示，可以看到页面左上角显示了登录用户的用户名“小叶子”。

若网站的所有页面都要求登录后才能访问，则应在所有的页面都加入上面这段代码。通过 7.4 节讲述的 include 指令很容易达到这一目的。



图 7-5　登录成功后的首页

7.3.4　重点提示

- 会话周期是指从客户端访问服务器开始，到会话结束的一段时间。
- session 对象供客户端在访问服务器的一次会话期间内使用，即客户端在这段时间内访问的所有网页都能读写服务器上属于自己的那个 session 对象。
- 每个客户端只能访问专属于自己的那个 session 对象。
- 向 session 对象中写入属性时，该属性值可以是一个 Java 对象，从 session 中取出该属性时，应强制还原为原来的类型。

7.4　公共页面的处理

前面提到，有的网站的所有页面都要求用户登录后才能访问，此时可以将强制登录的代码段单独写在一个文件中，然后通过页面包含指令在需要的页面中引用这个文件。

此外，网站的页面往往需要维持相同的风格，如不同的网页都有相同的 LOGO、导航条及版权信息等，通过 JSP 的包含指令或包含动作可以方便地实现这一目标。

7.4.1　include 指令

include 指令所实现的文件包含往往又称为“静态包含”，它用于通知 JSP 引擎在将 JSP 文件转换为 Servlet 时，将指定的资源内容内嵌到当前页面中该条指令出现的地方。这里的资源可以是 JSP 文件、HTML 文件，或者是其他类型的文本文件。

语法格式：

```
<%@ include file="relativeURL"/>
```

其中的 relativeURL 表示被包含的资源的文件路径（含文件名），若该路径以“/”开头，表示该路径是相对于网站根目录而言的，若不以“/”开头，则表示参照的是当前文件的位置。

【例 7.2】　include 指令的使用。

在页面 main.jsp 中使用 include 指令包含 in.jsp 的内容。

程序清单　web\7\7_4\in.jsp

```
1：<img src="me.jpg" />
2：<br/>
```

注意：创建该文件时，删除原来 IDE 自动生成的所有内容，然后添加上面这两行代码，即显示一张图片。

主要代码　web\7\7_4\main.jsp

```
1：<body>
2：　　测试文字 1<br/>
3：　　<%@include file="in.jsp" %>
4：　　测试文字 2
5：</body>
```

访问 main.jsp，得到如图 7-6 所示的运行结果。



图 7-6　include 指令的包含效果

使用 include 指令时还需注意以下两点。

- 被包含的页面若出现了设置页面编码的 page 指令，则主页面中所出现的编码设置必须与此完全相同，否则可能引起翻译时的错误。

■ include 指令所实现的包含操作是在当前的 JSP 文件被转换为.java 文件时进行的，不是在执行阶段进行的。因此，file 属性的值不能是变量，也不能以形如“head.jsp?aa=1”的形式向被包含页面传递参数，这些都与 7.4.2 节中的<jsp:include>动作所实现的动态包含有区别。

7.4.2　<jsp:include>动作

<jsp:include>是 JSP 的标准动作之一，用于在页面中动态地包含其他资源的执行结果。

在请求被执行时，若遇到<jsp:include>动作，则先执行指定的被包含文件，然后将执行结果包含进来。由于直到请求被执行时才能确定要包含的内容，因此使用<jsp:include>动作所实现的文件包含又被称为“动态包含”。

语法格式为：

```
<jsp:include page="relativeURL"/>
```

或

```
<jsp:include page="relativeURL">
    <jsp:param name="name" value="value" />
</jsp:include>
```

与 include 指令不同，这里的 page 属性值可以是一个表达式。

【例 7.3】　根据操作员的级别，包含不同的导航条。

include.jsp 中使用<jsp:include>动作包含导航条页面 nav.jsp，同时向被包含的页面传递参数 utype。

主要代码　文件 web\7\7_4\include.jsp

```
<body>
    <jsp:include page="nav.jsp">
        <jsp:param name="utype" value="admin"/>
    </jsp:include>
    <br/><br/><br/>
    主页面内容区域
</body>
```

nav.jsp 是被包含的导航条页面，代码如下：

主要代码　文件 web\7\7_4\nav.jsp

```
<body>
    <%
    String type = request.getParameter("utype");   //获取用户的类型
    if (type == null) {
        out.print("未知类型.");
        return;
    }
    if (type.equals("admin")) {
        out.print("<a href=\"dbadmin.jsp\" >后台管理</a>");
    } else {
```

```
11：                out.print("<a href=\"browse.jsp\" >商品浏览</a>");
12：            }
13：            %>
14：    </body>
```

根据主包含页面 include.jsp 向 nav.jsp 所传递的不同参数值，包含的内容也是不同的。因此，常把这种包含方式称为“动态包含”。

访问页面 include.jsp 的效果如图 7-7 所示。

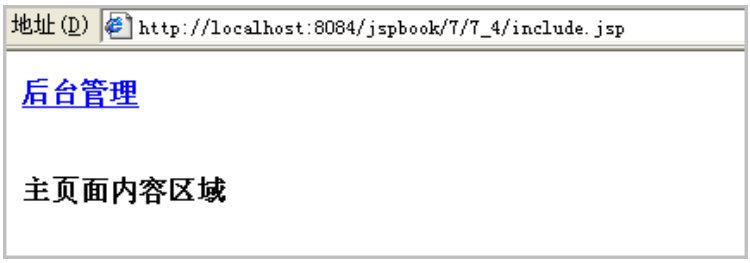


图 7-7　include 动作的执行效果

可改变 include.jsp 中<jsp:param>动作的参数值，查看实验的结果有何变化。

若不需向包含的页面传递参数，则应采用<jsp:include/>的形式，如:

```
<jsp:include page="head.jsp" />
```

注意结束前必须有斜杠“/”，表示标签结束。

7.4.3　任务实现

由于论坛的所有页面均采用相同的布局，因此可将页面头部的内容及页脚的内容各自编写为两个文件，然后在论坛网站的所有页面的相应位置包含这两个文件。下面演示了在登录页面中包含页头的情形。

【例 7.4】　在 head.jsp 中显示网站 LOGO 图片、导航条，并完成强制登录检测，在其他页面中可将此文件包含进来。

程序清单　web\7\7_4\head.jsp

```
1：<%@page pageEncoding="UTF-8"%>
2：<div id="logo">
3：    <img src="images/slogan.gif" />
4：</div>
5：<div id="nav">
6：    <div id="welcome">
7：        <%
8：        String uname = (String) session.getAttribute("uname");
9：        if (uname == null){
10：            response.sendRedirect("login.jsp");
11：            return;
12：        }
13：          out.print(uname+",欢迎进入 JSP 学习论坛");
```

```
14:            %>
15:        </div>
16:        <ul>
17:            <li><a href="index.jsp">首页</a></li>
18:            <li><a href="login.jsp">登录</a></li>
19:            <li><a href="reg.jsp">注册</a></li>
20:            <li><a href="mailto:jspbook@126.com">与我联系</a></li>
21:        </ul>
22: </div>
```

该文件单独运行的效果如图 7-8 所示。



图 7-8　页面头部

在 login.jsp 中使用包含指令“<%@ include file="head.jsp"/>”后的效果如图 7-9 所示。



图 7-9 在登录页面中包含了页头的效果

请大家模仿此例，将页脚的信息编写为文件 foot.jsp，然后在登录页面中包含进来。

7.4.4　重点提示

- 使用 include 指令或<jsp:include>动作均可包含其他文件的内容。
- include 指令只能静态包含一个文件的内容，而 j<sp:include>动作则可在执行时动态地包含一个页面的执行结果。
- <jsp:include>动作可以结合<jsp:param>动作向包含的页面传递参数。

7.5　在线会员名单的显示

7.5.1　application 对象的使用

application（应用）对象是 JSP 中的内建对象之一。在 session 对象中保存的属性只能被一个客户端所使用，且可以被该客户端所访问的不同网页共享，而在 application 对象中保存的属性则可以被不同的客户端在其访问本应用的整个过程中共享。

服务器启动且自动创建 application 对象以后，只要没有关闭服务器，application 对象将一直存在。由于 application 对象的生存期比较长，因此向 application 对象中写入数据之前，应考虑其必要性，请谨慎使用。

application 对象的常用方法如下。

- void setAttribute(String name,Object value)：向 application 中存入一个属性，名称由 name 指定，值由 value 指定。
- Object getAttribute(String name)：获取 application 中由 name 指定的名称的属性的值，若不存在，则返回 null。
- void removeAttribute(String name)：移除 application 中由 name 指定的属性。
- String getInitParameter(String name)：获取应用的初始参数值，该参数的名称由 name 指定。可以在网站的配置文件 web.xml 中设置网站的初始参数。
- String getRealPath(String path)：获取与虚拟路径对应的物理存储路径。

【例 7.5】 使用 application 对象发布信息。

当在一个客户端访问页面 inputMsg.jsp 中输入通知信息后，在 addMsg.jsp 中将该信息保存到 application 对象中，其他用户可以访问 showMsg.jsp 得到该通知的内容。

主要代码　文件 web\7\7_5\ex\inputMsg.jsp

```
1：<body>
2：        <form action="addMsg.jsp" method="POST">
3：          <table border="1" width="200">
4：            <caption>通知发布</caption>
5：              <tr>
6：                <td valign="middle">内容</td>
7：                <td><textarea name="msg" rows="5" cols="40">
8：                      </textarea>
9：                </td>
10：              </tr>
11：              <tr>
12：                <td colspan="2"><input type="submit" value="发布" /></td>
13：              </tr>
14：          </table>
15：        </form>
16：    </body>
```

主要代码　文件 web\7\7_5\ex\addMsg.jsp

```
1：    <body>
2：        <%
3：        request.setCharacterEncoding("utf-8");
4：        application.setAttribute("msg",request.getParameter("msg"));
5：        response.sendRedirect("showMsg.jsp");
6：        %>
7：    </body>
```

主要代码　文件 web\7\7_5\ex\showMsg.jsp

```
1：    <body>
```

```
2:          <%
3:          String msg=(String)application.getAttribute("msg");
4:          if (msg!=null)
5:             out.println("最新通知："+msg);
6:          else
7:             out.println("目前没有通知.");
8:          %>
9:       </body>
```

某一客户端访问 inputMsg.jsp，输入了如图 7-10 所示的通知内容。

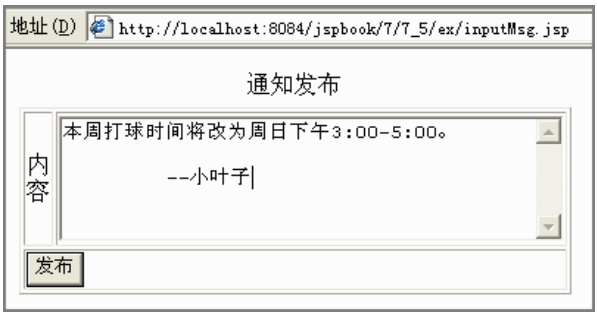


图 7-10　输入通知的内容

按下“发布”按钮提交后，可以看到发布的内容，如图 7-11 所示。

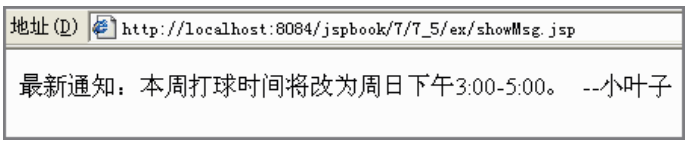


图 7-11　显示通知内容

若从其他客户端访问 showMsg.jsp，也将得到与图 7-11 相同的结果。

7.5.2　任务实现

要将在线用户的名单显示在页面上，应从以下几个方面完成。

- 在用户登录成功后，应将用户名保存到 application 对象中，可以向量的形式保存用户名单，然后将此向量保存到 application 中。
- 当用户登录成功后，页面导航条中原来的“登录”链接可改为“退出”链接。当用户单击了“退出”链接后，将用户名从名单列表中清除。
- 将在线名单显示在页脚区域，可将该区域的内容编写为一个单独的文件，而后在其他所有的页面中用指令或动作包含此文件。

复制 7.3 小节中得到的 chkLogin.jsp 文件，将其登录成功后的代码做如下修改：

修改后的新代码　web\7\7_5\chkLogin.jsp

```
1: if (valid) {
2:     session.setAttribute("username",uname);
3:     Vector visitors = (Vector) application.getAttribute("namelist");
```

```
4:      if (visitors == null) {
5:          visitors = new Vector();
6:          application.setAttribute("namelist", visitors);
7:      }
8:      boolean unique=true;
9:      for (int i=0;i<visitors.size();i++)
10:         if (((String)visitors.get(i)).equals(uname))
11:             {unique=false;break;}
12:     if (unique) visitors.add(uname);
13:     out.println("<br/>登录成功!正在自动转向网站首页...");
14:     response.setHeader("refresh", "3;url=index.jsp");
15:   }
```

代码说明如下。

- 第 3 行从 application 对象中取出名称为 namelist 的属性，保存到向量 visitors 中。
- 第 4～7 行判断向量 visitors 是否为空对象，即取出的属性是否存在，若无此属性，则实例化一个新的向量，并将它以 namelist 的名称写入 application 对象中。
- 第 8～11 行通过遍历向量 visitors，检查名单列表是否已存在刚登录的用户名，避免因为用户重复登录而致使名单中出现多个重复的用户名。若未重复，则在第 12 行将新用户名写入向量中。
- 其余代码未做修改，这里不再说明。

编写显示在线名单的页面 namelist.jsp，代码如下：

程序清单 web\7\7_5\namelist.jsp

```
1: <%@page contentType="text/html" pageEncoding="UTF-8"%>
2: <%@page import="java.util.*" %>
3: <%
4:      String name = null, namelist = "";
5:      Vector visitors = (Vector) application.getAttribute("namelist");
6:      if (visitors != null) {
7:         Iterator it = visitors.iterator();
8:         while (it.hasNext()) {
9:            name = (String) it.next();
10:           namelist += name + " ";
11:        }
12:        if (!namelist.equals("")) {
13:           out.println("<p>在线会员：" + namelist+"</p>");
14:        }
15:     }
16: %>
```

代码说明如下。

- 第 5 行从 application 对象中取出名单列表。
- 第 6 行判断该名单是否为空。
- 第 7～11 行，遍历名单表，将其中所有的用户名取出后连接成一个字符串。
- 第 12～14 行，若得到的名单字符串非空，则显示在线名单。

编制显示页脚的公共页面 foot.jsp，代码如下：

程序清单　web\7\7_5\foot.jsp

```
<div id="footer">
    <jsp:include page="namelist.jsp" />
    <span>
    2008-2010 &copy; V-Sunny Technology Company. All rights reserved.
</span>
</div>
```

在论坛的其他显示页面，可以通过 include 指令将该页脚文件包含进来。

修改页头文件 head.jsp，判断用户是否登录，若已登录，则将原来的“登录”链接修改为“退出”链接。导航条部分的代码修改如下：

修改后的导航条代码　web\7\7_5\head.jsp

```
<ul>
            <li><a href="index.jsp">首页</a></li>
            <% if (session.getAttribute("username") == null) {
            %>
            <li><a href="login.jsp">登录</a></li>
            <%} else {%>
            <li><a href="exit.jsp">退出</a></li>
            <%}%>
          <li><a href="reg.jsp">注册</a></li>
           <li><a href="mailto:jspbook@126.com">与我联系</a></li>
</ul>
```

在论坛的其他显示页面中，应将页头文件 head.jsp 和页脚文件 foot.jsp 的内容包含进来，如修改后主页面 index.jsp 的代码如下所示：

主要代码　web\7\7_5\index.jsp

```
<div id="container">
    <%@include file="head.jsp" %>
    <!--下面为 content 层，在这个页面里，这个层用来显示首页的主要内容 -->
    <div id="content">
       内容区域
    </div> <!--end of content-->
    <%@include file="foot.jsp" %>
</div> <!--end of container-->
```

最后，编写退出页面 exit.jsp，当用户单击了“退出”按钮时，将当前用户名从在线名单中清除，代码如下：

主要代码　web\7\7_5\exit.jsp

```
<body>
    <%
      Vector visitors=(Vector)application.getAttribute("namelist");
      visitors.remove(session.getAttribute("username"));
      session.removeAttribute("username");
```

```
6:      response.sendRedirect("login.jsp");
7:      %>
8:  </body>
```

代码说明如下：

- 第 3 行从 application 对象中取出在线名单。
- 第 4 行从名单中清除当前用户的用户名。
- 第 5 行从 session 对象中清除用户名。
- 第 6 行重定向到登录页面。

将以上文件复制到 jspbook 项目的 web 目录下，访问 login.jsp，输入用户名“小叶子”，提交后得到如图 7-12 所示的运行结果。



图 7-12　登录成功后在主页显示的在线会员名单

当我们再次访问登录页面，或从其他客户端以另一个用户名“sunny”登录时，将在页脚看到目前在线的所有用户名单，如图 7-13 所示。



图 7-13　多个用户登录后的在线名单

若单击页面的“退出”链接后，再查看登录页面的页脚，此时显示的在线名单又和图 7-12 所示的结果一样了。

其实，这里只是大致地显示了用户的在线情况，若用户未单击“退出”链接退出，该例将无法从在线名单中清除其用户名。一般地，可以结合 session 的监听器，呈现更为精确的在线名单。

7.5.3　重点提示

- 使用 application 对象保存的属性可供访问该网站的所有客户端共享。
- 在 application 中保存的属性是一个对象，取出时为 Object 类型，应强制还原为写入时的类型。

7.6　使用 Cookie 完成自动登录功能

7.6.1　任务提出

有时，用户希望登录一次后，在一段时间内都能免去登录的步骤，即网站可提供自动登录的功能。通过 Cookie 机制可实现这一需求。

7.6.2　Cookie 的使用

Cookie 是一种能够让 Web 服务器把少量数据储存到客户端的硬盘或内存，之后又能从客户端读回数据的一种技术。

Cookie 的作用与 JSP 的内置对象 session 有些相似，都可以保存某个客户端的信息。但是，Cookie 将信息保存在客户端，而 session 却将信息保存在服务器上。由于 Cookie 的信息往往保存在硬盘上，因此它保存信息的时间通常比 session 更长。

在 Windows 2000/XP 的计算机中，Cookies 文件的存放位置为“C:/Documents and Settings/用户名/Cookies”目录中。由于 Cookie 信息一般是以明文格式保存的，因此使用 Cookie 也存在一定的安全问题，因此客户端有可能会禁用 Cookie。

Java 中使用 Cookie 类的实例来存储 Cookie 信息，但在 JSP 中对 Cookie 的读写却是通过 request 对象和 response 对象实现的。

使用 Cookie 的基本方法如下。

1．创建 Cookie 对象

```
Cookie c=new Cookie("username","sunny");
```

该构造方法的两个参数依次表示 Cookie 的名称和值，这里的值只能是字符串类型。

2．设置 Cookie 的有效时间

```
c.setMaxAge(30*24*60*60);
```

时间的单位为秒，当该时间数为 0 时，表示删除 Cookie，为负值时则在关闭浏览器后删除 Cookie。

3．将 Cookie 信息发往客户端

```
response.addCookie(c);
```

4．读取 Cookie 信息

```
Cookie cks[]=request.getCookie();
String name=cks[0].getName();
String value=cks[0].getValue();
```

从 request 中读取到的 Cookie 信息是一个由 Cookie 对象组成的数组，包含了来自客户端的属于本网站的所有 Cookie 信息。对于保存在 Cookie 中的一项数据，可使用 getName 方法取得它的名称，而使用 getValue 方法则可取得它的值。

向客户端写入 Cookie 信息时，利用了内置对象 response 的 setCookie 方法，当浏览器接收到服务器的响应信息时，若其中有 Cookie 信息，则将其写入客户机的硬盘或内存中。

而从客户端读取 Cookie 信息时，则利用了内置对象 request 的 getCookies 方法。事实上，当客户端向服务器发送一个请求时，浏览器会从自己的计算机上寻找该服务器存入的 Cookie 信息。若找到，则和其他请求信息一道被发送到服务器，服务器则将来自客户端的所有信息封装在 request 对象中。

【例 7.6】　Cookie 的写入与读取。

页面 setCookie.jsp 将用户名保存到客户端的 Cookie 中，保存期为 30 天，而页面

getCookie.jsp 则读取保存在客户端 Cookie 中的用户名。

这两个页面的主要代码如下。

主要代码　setCookie.jsp:

```
1：<%
2：    Cookie c=new Cookie("username","Helen");
3：    c.setMaxAge(30*24*60*60);   //Cookie 保存的时间，单位为秒
4：    response.addCookie(c);
5：%>
```

第 3 行代码设置了 Cookie 的保存期限，单位为秒；若未设置该期限，则此 Cookie 信息只在会话期间有效，当用户关闭浏览器时，Cookie 自动失效。

若需要删除某个 Cookie，只需将其有效期设为 0 即可。

主要代码　页面 getCookie.jsp

```
1：<%
2：Cookie cookie[] = request.getCookies();
3：if (cookie == null) out.print("读不到 cookie");
4：else     for (int i = 0; i < cookie.length; i++)
5：   { if (cookie[i].getName().equals("username"))
6：          out.print(cookie[i].getName()+":"+cookie[i].getValue());
7：         }
8：%>
```

由于保存在客户机上的 Cookie 信息可能有多个，因此利用 request 对象得到所有的 Cookie 对象的数组后，需要遍历这个数组，检查每个 Cookie 对象中的数据名称，才知道哪个是自己需要的信息。

需要注意的是，若保存的信息是中文的，则需用 java.net.URLEncoder 类的 encode 方法和 java.net.URLDecoder 类的 decode 方法进行编码、解码。请大家自行进行测试中文 Cookie 信息的保存。

7.6.3　任务实现

要完成网站的用户自动登录功能，应完成以下 3 个任务。

- 需要修改登录页面 login.jsp，在其中添加选择保存期限的选项。
- 需要修改登录信息的检测页面 chkLogin.jsp，若用户选择了自动登录选项，则在登录成功后按照用户的选择期限将用户的信息保存到 Cookie 中。
- 需要修改页头文件 head.jsp 中的强制登录检测代码，当用户未登录时，应转而检查该客户端的 Cookie 信息，以确定是否可采用自动登录方式。

1．修改登录界面

复制 7.4 节得到的 login.jsp，在页面中表单内恰当的位置添加一组单选按钮，以提供 Cookie 保存期限的选择，相关代码如下所示：

```
<input type="radio" name="limit" value="0" checked="checked">不保存
<input type="radio" name="limit" value="30" />一个月
<input type="radio" name="limit" value="180" />半年
```

每个单选按钮标签中的 value 值表示保存 Cookie 的天数，第 1 个选项为默认值，保存天数为 0。

2．修改 chkLogin.jsp 页面

复制 7.5 节的 chkLogin.jsp 文件，在登录成功并且完成在线名单的处理后，即代码“out.println("
登录成功!正在自动转向网站首页...");”的前面，添加如下的 Cookie 处理代码：

添加的代码　web\7\7_6\chkLogin.jsp

```
1:    int limit = Integer.parseInt(request.getParameter("limit"));
2:    Cookie c = new Cookie("username",
      java.net.URLEncoder.encode(request.getParameter("uname"),"UTF-8");
3:    c.setMaxAge(limit * 24 * 60 * 60);
4:    response.addCookie(c);
```

代码说明如下。

- 第 1 行读取用户选择的 Cookie 保存天数。
- 第 2 行创建一个新的 Cookie 对象，名称为 username，取值为用户所输入的用户名，由于用户名可能为中文，因此需要先使用 URLEncoder 类的静态方法 encode 对其进行编码。
- 第 3 行根据用户输入的天数，设置 Cookie 的有效期限。若用户选择的是不保存，则天数为 0，将删除相应的 Cookie 信息。
- 第 4 行使用 response 对象的 addCookie 方法，将 Cookie 对象的信息发往客户端。

3．修改 head.jsp

修改强制登录的处理代码，当用户未登录时，若能在客户端的 Cookie 信息中检测到有自动登录的设置，则应为其完成自动登录。复制上一节的 head.jsp 文件，寻找下面的代码段：

```
if (uname == null) {
   response.sendRedirect("login.jsp");
   return;
}
```

在上面这段代码前添加如下的内容：

添加的代码　web\7\7_6\head.jsp

```
1：if (uname == null) {
2：    Cookie ck[] = request.getCookies();
3：      if (ck != null) {
4：         for (int i = 0; i < ck.length; i++) {
5：            if (ck[i].getName().equals("username")) {
6：              uname = ck[i].getValue();
7：                session.setAttribute("username",
        java.net.URLDecoder.decode(uname,""UTF-8));
                  break;
8：            }
9：         }
10：      }
11：    }
```

代码说明如下。

- 第 1 行判断从 session 中取出的用户名是否为空对象，若为 null，则表示用户未登录。
- 第 2 行从 request 对象中取出所有的 Cookie 对象，保存到数组 ck 中，该数组的每一个元素都是一个 Cookie 对象，各表示一项 Cookie 信息。
- 第 3 行判断得到的 Cookie 数组是否为空，若空，则表示没有 Cookie 信息。
- 第 4～10 行通过 for 循环语句检查 Cookie 数组中的每一个元素，若其名称为 username，则取出其值，将其保存到 session 中，这标志着用户已经登录了，并且在第 7 行中止循环，停止检查剩余的 Cookie 信息。

将以上文件复制到 jspbook 项目的 web 目录中，访问登录页面 login.jsp，运行效果如图 7-14 所示。



图 7-14　在登录页面选择 Cookie 保存期限

选择在“一个月”内自动登录，提交后的运行结果和 7.5 节相同。关闭此浏览器窗口，访问 index.jsp 页面，会发现该页面中已经显示了用户的用户名，这表明程序已经自动完成了登录。

再次访问登录页面 login.jsp，并且将其中的保存期限选择为“不保存”。登录后，关闭浏览器窗口，结束会话，然后重新访问 index.jsp 页面，此时会发现被自动重定向到了登录页面。这表明 Cookie 信息已被删除，因此无法自动登录了。

7.6.4　重点提示

通过 Cookie 机制或 session 对象均可保存某客户端的信息，但二者之间存在着明显的差别。

- Cookie 信息保存在客户端的硬盘或内存中，而 session 对象保存在服务器的内存中。
- 在 Cookie 中保存的信息都是字符串，而保存在 session 中的信息可以是其他类型的对象。
- Cookie 通常用来较长时间地保存客户端的信息。
- Cookie 不是 JSP 的内置对象。



思考与训练

一、基础训练

请按要求写出相应的代码，各类数据库服务器的运行端口均保持默认值，未特别指出服务器 IP 或名称的，均为本地机器。

1．连接 IP 地址为 172.33.6.28 上的 Oracle 服务器，实例名称为 orcl，连接服务器的用户名为 soy，密码为 bean888。

2．连接 SQL Server 2005 数据库，数据库名称为 tb，用户名为 cat，密码为 miao。

3．连接 MySQL 数据库，数据库名称为 cinema，用户名为 root，密码为 123。

4．请分别使用 include 指令和 include 动作，在页面中包含 head.jsp。

5．将字符串变量 username 的值保存到 session 中，属性名称为 uname。

6．将 ArrayList 类的对象 goods 保存到 session 中，属性名称为 cart。

7．从 session 中取出 ArrayList 类型的参数 goods，保存到对象 g1 中。

8．从 session 中取出 Integer 类型的参数 total 的值，并赋值给 int 型变量 t。

9．判断 session 中是否存在参数 username，若不存在则直接跳转到 login.jsp。

10．判断 session 中是否存在参数 username，若不存在则显示文字“您尚未登录”，并在 3 秒后跳转到 login.jsp。

二、思考题

1．include 指令与<jsp:include>动作有何异同？

2．若页面 a 包含了页面 b，在页面 a 中定义了变量 n，能否在页面 b 中使用？请分别说明使用 include 指令和动作的情形。

3．什么是一个会话周期？何时开始、何时结束？

4．session 对象一般用于什么场合？试举例。

三、综合实训

1．在 MySQL 或其他数据库中创建论坛的版块信息表，编写 JSP 页面，完成版块的增、删、查、改功能。

2．完成论坛的用户退出功能。

3．创建商品信息表，编写 JSP 页面实现商品选购及显示选购清单的功能。

第 8 章　帖子浏览功能的实现

8.1　显示所有帖子的概要信息

8.1.1　任务提出

当论坛的用户完成登录后，将转到论坛的首页。在这里，帖子会按最后更新的时间排序，以列表的形式显示帖子的标题、作者、最后发表/回复时间、点击数/回复数等信息，最后更新的帖子排在最前面，单击帖子的标题后，应转向显示本帖子的内容及回复信息的详情页面；若用户未登录，则自动转向登录页面。

该功能主要通过数据库查询技术来实现，当从数据库中查询得到所需的数据后，可使用 JSP 表达式的形式将数据显示在页面上。

8.1.2　JSP 的表达式

在 JSP 中，可使用符号“<%=”和“%>”括起一个表达式，作用是计算并输出该表达式的值。该表达式在页面文件中的位置，也就是其计算结果所显示的位置。注意“<%=”这 3 个字符之间不能有空格，且符号“%>”前面没有分号“;”。

例如：

```
<%= a+b %>
```

将输出 a+b 的结果，相当于代码 <% out.print(a+b);%> 的效果。

在实际应用中，表达式往往会结合其他形式的 JSP 页面元素一起使用。

■ 嵌入在 HTML 代码中：

```
<a href="list.jsp?pageno=<%=page+1%>">下一页</a>
```

这里计算出 page+1 的结果，并作为参数 pageno 的值提交给 list.jsp 页面。

■ 嵌入在 JSP 的动作中：

```
<jsp:setProperty name="u1" property="uname" value="<%=myname%>" />
```

这个动作用以设置 JavaBean 实例的某个属性的值，第 10 章将介绍 JavaBean 的相关知识。

8.1.3　任务实现

要显示帖子标题等概要信息，主要是查询主题帖表，并将各条记录相关列的信息显示在页面上，代码如下所示：

主要代码　web\8\8_1\index.jsp

```
1：<%@page contentType="text/html" pageEncoding="UTF-8"
 import="java.sql.*"%>
2：<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```

```
3:    "http://www.w3.org/TR/html4/loose.dtd">
4:
5: <html>
6:    <head>
7:       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
8:       <title>JSP 学习论坛---首页</title>
9:          <style type="text/css">
10:          @import url("css/common.css");
11:       </style>
12:    </head>
13:    <body>
14:    <div id="container">
15:       <%@ include file="head.jsp"    %>
16:       <div id="content">
17:          <div id="postlist">
18:             <div id="newPostLink">
19:             <a href="newPost.jsp">
20:                 <img src="images/newtopic1.gif" border="0"/>
21:             </a>
22:           </div>
23:    <%
24:    String driverClass = "org.gjt.mm.mysql.Driver";
25:    String url = "jdbc:mysql://localhost:3306/bbs
                ?useUnicode=true&characterEncoding=utf8";
26:    String username = "root";
27:    String password = "rootroot";
28:    String title, date, author, rauthor, rdate;
29:    int replies, clicks, pid;
30:    Connection con = null;
31:    Statement st = null;
32:    ResultSet rs = null;
33:    try {
34:       String sql = "select * from posts order by plastUpdate desc";
35:       Class.forName(driverClass);
36:       con = DriverManager.getConnection(url, username, password);
37:       st = con.createStatement();
38:       rs = st.executeQuery(sql);
39:       if (rs != null) {
40:          out.print("<table id='pListTable' cellspacing='0'>");
41:          out.print("<tr class='head'><th>主  题</th><th>作  者</th>
              <th>最后更新</th><th>回复数</th><th>单击数</th></tr>");
42:        /*---- rs.next()可以把结果集的游标移到下一条  ----*/
43:          while (rs.next()) {
44:             /*下面开始把当前这条记录的各个字段值提取出来*/
45:             pid = rs.getInt("pid");          //获取该条记录的帖子 ID
46:             title = rs.getString("ptitle");      //获取帖子主题
```

```
47:             date = rs.getString("pdate") == null ? "" :
        rs.getString("pdate").substring(2, 16);//获取发表日期
48:             author = rs.getString("pauthor");//获取发布人
49:             replies = rs.getInt("preplies"); //获取回复数
50:             clicks = rs.getInt("pclicks"); //获取点击数
51:             rdate = rs.getString("plastUpdate") == null ? "" :
        rs.getString("plastUpdate").substring(2, 16); //最后更新时间
52:             rauthor = rs.getString("plastAuthor");   //最后作者
53:         %>
54:         <!--下面在表格的一行里显示帖子的信息-->
55:         <tr>
56:           <td class="title" >
57:             <img src="images/hot_red_folder.gif" />  
58:             <a href="postDetail.jsp?pid=<%=pid%>" title="点击进入">
59:                 <%=title%>
60:             </a>
61:           </td>
62:           <td><%=author%><br><%=date%></td>
63:           <td><%= (rauthor == null ? "---" : rauthor) + "<br>" +
          (rdate == null ? "---" : rdate)%></td>
64:           <td><%=replies%></td>
65:           <td><%=clicks%></td>
66:         </tr>
67:         <%
68:           }//end of while，所有帖子都显示完毕
69:           out.print("</table>");//表格显示完毕
70:         }
71:       } catch (SQLException e) {
72:         out.println("网站出错，请与管理员联系.");
73:         e.printStackTrace(System.err);
74:       } finally {
75:         try {
76:           if (rs != null) {
77:             rs.close();
78:           }
79:         } catch (Exception e) {
80:           e.printStackTrace(System.err);
81:         }
82:         try {
83:           if (st != null) {
84:             st.close();
85:           }
86:         } catch (Exception e) {
87:           e.printStackTrace(System.err);
88:         }
89:         try {
```

```
90：                if (con != null) {
91：                    con.close();
92：                }
93：            } catch (Exception e) {
94：                e.printStackTrace(System.err);
95：            }
96：        }
97：    %>
98：            </div>
99：        </div> <!--end of content-->
100：        <%@include file="foot.jsp" %>
101：    </div>
102：  </body>
103：</html>
```

代码说明如下。

- 第 15 行和第 100 行分别使用 include 指令包含了页头和页脚文件，这两个文件的内容与第 7 章 7.6 节中的同名文件相同。由于在 head.jsp 页面中要求用户必须登录，故在访问本页面时，若未登录，则会自动转去登录页面，待登录成功后，才自动转回首页。
- 第 18～22 行提供了发表新帖的超链接。
- 第 24～27 行分别定义了连接数据库时所需的驱动类名、URL、用户名及密码。
- 第 28～29 行定义了几个变量，分别用于表示帖子的标题、作者、发布时间、最后作者、最后更新时间及回复数、点击数和帖子的 ID。
- 第 30～32 行定义了完成数据库查询操作时所需要的连接对象、语句对象及结果集对象。
- 第 34～38 行完成数据库查询，并将结果集保存到对象 rs 中。
- 第 39 行判断结果集是否非空，若非空，则通过第 43～69 行遍历结果集，并以表格的形式在页面上输出各条记录的内容；其中第 55～66 行在表格的一行里显示了一条帖子的信息，主要采取了表达式的形式显示各项数据。
- 第 74～96 行逆序关闭各项数据库资源。

页面的运行效果如图 8-1 所示。

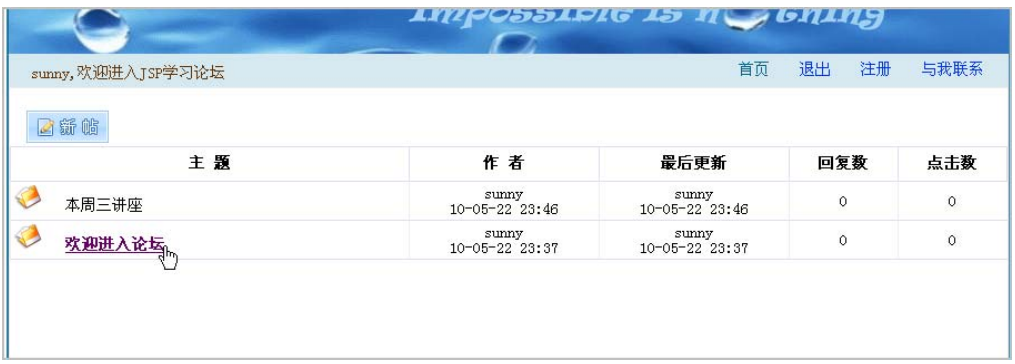


图 8-1　论坛首页中帖子列表的显示

帖子详情显示页面的实现过程与本例相似，主要工作也是查询数据库，只是该页面中需显

示作者的基本信息、主题帖和相关回复帖的信息，因此需要访问 3 张表。为了简化代码的编写，可先在数据库中创建相关的视图，请读者尝试完成该页面的编写工作。

8.1.4　重点提示

- 表达式“<%=E%>”与代码“<% out.print(E);%>”具有相同的作用。
- 注意表达式的结束符“%>”前面没有分号。

8.2　搜 索 帖 子

8.2.1　任务提出

当论坛经过一段时间的使用后，帖子的数量在逐渐增多，为了使访问者能尽快查找到自己感兴趣的帖子，提供帖子的搜索功能是非常必要的。因此，本节将修改前面得到的帖子列表显示页面，加入按标题或作者关键字搜索帖子的功能。

8.2.2　自提交的表单

有时，当用户在页面上输入一些信息后，仍提交到原来的页面进行处理，这种将信息提交给原页面的表单可称为“自提交”的表单。此时，页面可以根据请求对象 request 中是否存在表单数据，来判断用户对该页面的访问发生于表单数据提交前还是提交后，以便采取不同的处理。

【例 8.1】　自提交的表单页面。

当用户在页面 selfSubmit.jsp 中输入暗号后，提交到原来的页面，系统仍会在该页面提取用户提交的数据，然后显示在浏览器中。

主要代码　web\8\8_2\selfSubmit.jsp

```
1:  <body>
2:    <%
3:        if (request.getParameter("msg") == null) {
4:    %>
5:    <form action="" method="post">
6:      暗号<input type="text" name="msg" />
7:      <input type="submit" value="提交" />
8:    </form>
9:    <%} else {
10:         request.setCharacterEncoding("UTF-8");
11:         String msg=request.getParameter("msg");
12:         out.println("您输入的暗号是：" + msg);
13:       }
14:    %>
15:  </body>
```

代码说明如下。

- 第 3 行判断是否提交了名称为 msg 的请求参数，若为 null，则表示未提交该参数，因

此通过第 5～8 行输出一个表单，用户可在此输入暗号信息，第 5 行 form 标签的 action 属性取空字符串，则表示提交给表单所在的页面（即 selfSubmit.jsp），实际上也可省略不写该属性。

- 若用户已经提交了名称为 msg 的请求参数，则通过代码第 10～12 行提取用户的输入信息，并显示在页面上。

该页面的运行效果如图 8-2 所示。

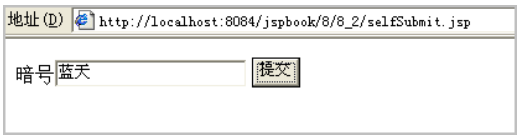


图 8-2　在自提交的页面中输入信息

提交后的结果如图 8-3 所示。

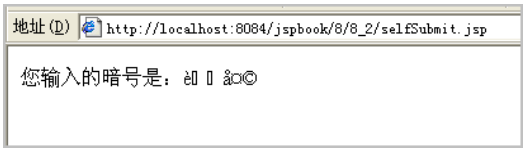


图 8-3　输出中文信息乱码

当使用自提交这种方式时，可能会发生下面的情形：即使在提取信息前已经使用 request 对象的 setCharacterEncoding 方法设置了请求参数的编码字符集，读取中文信息时仍会发生乱码的现象，此时可采取其他方式解决这个中文编码的问题。将程序第 10～12 行代码更改为下面的代码段：

```
String msg= request.getParameter("msg");
msg=new String(msg.getBytes("iso8859-1"),"UTF-8");
out.println("您输入的暗号是：" + msg);
```

代码“msg=new String(msg.getBytes("iso8859-1"),"UTF-8");”表示将得到的表单数据 msg 按 iso8859-1 编码还原成字节数组，再按 UTF-8 编码重新解释，组合成新的字符串。

修改后再次运行，将得到正确的显示结果，如图 8-4 所示。



图 8-4　在自提交页面正确读取中文信息

8.2.3　任务实现

要在帖子列表显示页面添加条件搜索，只需要在页面上添加条件输入表单，并添加对提交的输入条件的处理即可。下面对前面得到的 index.jsp 页面进行修改。

1．添加条件输入表单

在原来的“发表新帖”超链接后面添加表单代码，如下所示：

```
1：<div id="newPostLink">
2：<a href="newPost.jsp"><img src="images/newtopic1.gif" border="0"/></a>
3：</div>
4：<div id="searchDiv">
5：    <form method="post" name="search">
6：        按<select name="keyField">
7：            <option value="ptitle" selected>标题</option>
8：            <option value="pauthor">作者</option>
9：        </select>
10：           关键字<input type="text" name="keyValue" />
11：         <input type="submit" name="submit" value="搜索帖子" />
12：    </form>
13：</div>
```

上面的代码段中，第 1～3 行是原来程序中已有的发表新帖的超链接，从第 4 行开始至第 13 行为新添加的表单层代码。这里添加了层 searchDiv，相应地，在 CSS 文件中需添加对该层的定位控制代码。

2．根据输入条件构造带条件的查询语句

在代码“String driverClass = "org.gjt.mm.mysql.Driver";”的前面添加下面的代码段：

```
1：//确定搜索条件
2：  String searchStr="";
3：  if (request.getParameter("keyValue")!=null){
4：      String keyField=request.getParameter("keyField");
5：      String keyValue=new String(request.getParameter("keyValue")
    .getBytes("iso8859-1"),"utf-8");
6：        searchStr=" where "+keyField+" like '%"+keyValue+"%'";
7：  }
```

代码说明如下：

- 字符串变量 searchStr 表示查询数据时所带的 where 子句，初始化为空串，即查询时无条件限定。
- 第 3 行判断是否有提交查询条件，若有，则通过第 4、5 行分别取出条件查询的字段名和关键字的值，由于关键字可能含有中文，因此第 5 行将提取到的关键字值按 utf-8 重新编码。
- 第 6 行根据得到的字段名和关键字值构造出相应的 where 子句。

最后，将原来的查询语句代码

```
String sql = "select * from posts order by plastUpdate desc";
```

更改为：

```
String sql = "select * from posts "+searchStr+" order by plastUpdate desc";
```

即在查询语句中添加进行条件限定的 where 子句，由字符串 searchStr 表示。

访问 index.jsp 页面，得到如图 8-5 所示的结果。

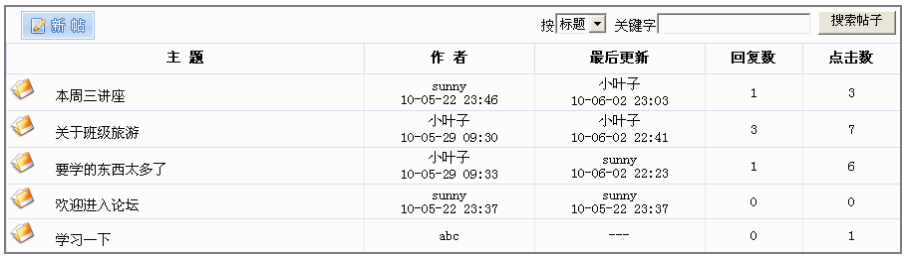


图 8-5　未进行条件限定时的帖子列表

当在关键字输入框中输入“学”时，得到如图 8-6 所示的结果。

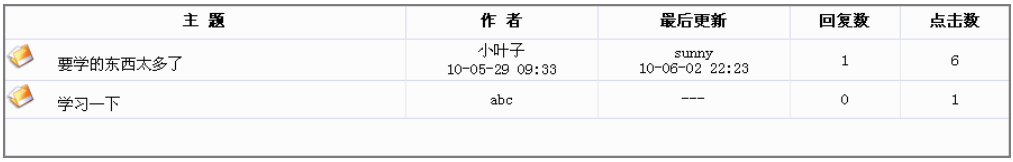


图 8-6　按标题关键字搜索帖子

8.3　分 页 显 示

8.3.1　任务提出

在前面的帖子列表显示页面，所有的帖子都显示在一个页面中，当论坛的帖子非常多时，用户浏览起来会十分不便。因此，必须将帖子列表进行分页显示，以方便用户浏览。

8.3.2　分页显示的原理

数据的分页显示需要完成两方面的工作：一是显示指定页的数据；二是计算记录的条数，从而提供相应的页号链接导航。

要显示指定页的记录，比较简单的方法是，先查询出所有符合条件的记录集合，再根据页号计算该页的第一条记录的行号，而后利用 ResultSet 接口的 absolute 方法（即绝对定位）将游标移向该条记录，接着显示从这条记录开始的一页数据。这种做法的优势是，完全使用 JDBC 去实现分页，能适应不同类型的数据库，处理过程也比较简单，但这种做法需将表中所有的记录都查询出来，效率比较低下。

效率较高的做法是，构造高效的查询语句，使其只查询出一页的记录。对于不同类型的数据库，该查询语句的写法有所不同。以表 table1 为例，设该表的主键字段为 tid，每页显示 10 条记录，下面给出了常见类型的数据库取第 3 页记录的 SQL 语句的写法。

1．SQL Server 2000、SQL Server 2005 或 Access

在这些数据库类型中，取表中前 10 条记录的 SQL 语句为：

```
select top 10 * from table1
```

若要取第 3 页的记录，可以先取出前 30 条记录，然后去除最前面的 20 条，剩下的自然就

是第 3 页的 10 条记录了。根据这样的思路，得到下面的 SQL 语句:

```
select top 10 * from table1 where tid not in (select top 20 tid from table1)
```

这里利用子查询去除了前面两页的记录。

2．Oracle

Oracle 数据库中执行的 SQL 语句不能使用 top 关键字。但是，从 Oracle 查询得到的记录集中，每条记录都有一个虚拟列，即行号 rownum，利用这一特点，得到取 Oracle 的数据表中第 21～30 条记录的 SQL 语句如下：

```
select * from (select *,rownum num from table1 where rownum<=30) where num>=20;
```

这里需要注意的是，子查询中必须给 rownum 列另起一个别名。

3．MySQL

MySQL 中有一个关键字 limit，使用它可以方便地获取查询结果的子集。取出第 3 页的 10 条记录的 SQL 语句为：

```
select * from table1 limit 20,10
```

limit 子句的格式为：

```
limit [起始行号,]提取的条数
```

若省略起始行号，则表示从第 1 条记录开始取，第 1 条记录的行号为 0。

8.3.3　任务实现

下面对 8.2 节得到的 index.jsp 页面进行修改，得到分页显示的帖子列表显示页面。该页面的运行效果如图 8-7 所示。

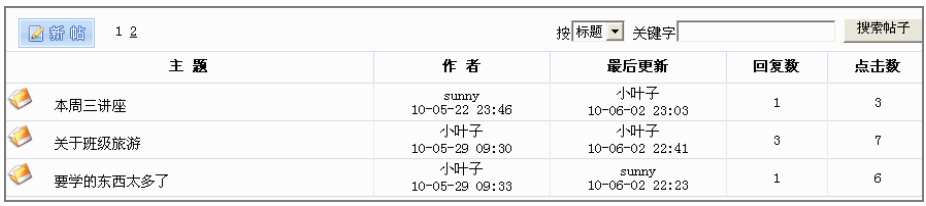


图 8-7　分页显示中第 1 页的显示结果

可以看到在“新帖”链接后面，显示了页号链接，当单击页号“2”后，得到如图 8-8 所示的结果。

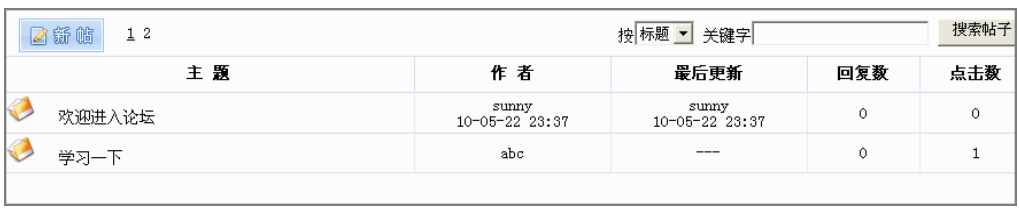


图 8-8　分页显示中第 2 页的显示结果

由于测试时主题帖表的记录条数不多，为了展示分页的效果，这里将每页显示的记录条数设为 3 条，实际使用时可自行更改。

主要代码如下所示：

主要代码　文件 web\8\8_3\index.jsp

```
<body>
    <% //确定搜索条件
        String searchStr = "";
        if (request.getParameter("keyValue") != null) {
            String keyField = request.getParameter("keyField");
            String keyValue = new String(request.getParameter("keyValue").
        getBytes("iso8859-1"), "utf-8");
            searchStr = " where " + keyField + " like '%" + keyValue + "%'";
        }
        //获取页号
        String spageno = request.getParameter("pageno");
        if (spageno==null) spageno="1";
        int pageno = 1;   //默认页号为 1
        try {
            pageno = Integer.parseInt(spageno);
        } catch (Exception e) {
            e.printStackTrace(System.err);
        }
        //创建数据库连接
        String driverClass = "org.gjt.mm.mysql.Driver";
        String url = "jdbc:mysql://localhost:3306/bbs?
        useUnicode=true&characterEncoding=utf8";
        String username = "root";
        String password = "rootroot";
        Connection con = null;
        Statement st = null;
        ResultSet rs = null;
        try {
            Class.forName(driverClass);
            con = DriverManager.getConnection(url, username, password);
            st = con.createStatement();
        } catch (Exception e) {
            e.printStackTrace(System.err);
            out.println("网站出错，请与管理员联系.");
            return;
        }
        //计算页号导航
        int allpage = 0, allrec=0,pagesize = 3;     //每页显示 3 条记录
        rs = st.executeQuery("select count(*) as total from posts "
        + searchStr);
        if (rs != null && rs.next()) {
            allrec = rs.getInt(1);
        }
```

```
41:             allpage=allrec%pagesize==0?allrec/pagesize:allrec/pagesize+1;
42:             int firstp = 1, lastp = allpage;
43:             if (pageno - 3 > 1) {
44:                firstp = pageno - 3;
45:             }
46:             if (pageno + 3 < lastp) {
47:                lastp = pageno + 3;
48:             }
49:       %>
50: <div id="container">
51:    <%@ include file="/head.jsp"    %>
52:    <div id="content">
53:       <div id="postlist">
54:          <div id="newPostLink">
55:             <a href="newPost.jsp">
56:               <img src="images/newtopic1.gif" border="0"/>
57:          </a>
58:         </div>
59:         <div id="pageNav">
60:           <%
61:                if (firstp > 1) {
62:           %>
63:           <a href="index.jsp?pageno=1">首页</a>
64:           <%}
65:             for (int i = firstp; i <= lastp; i++) {
66:                if (i == pageno) {
67:                   out.println(i);
68:                } else {
69:                   out.println("<a href='index.jsp?pageno=" + i + "'>"
              + i + "</a>");
70:                }
71:             }
72:                if (lastp < allpage) {
73:          %>
74:          <a href="index.jsp?pageno=<%=allpage%>">末页</a>
75:          <%}%>
76:       </div>
77:       <div id="searchDiv">
78:            <!--查询条件输入表单，代码同前一小节，略 -->
79:       </div>
80:       <div style="clear:both">
81:          <%
82:            String title, date, author, rauthor, rdate;
83:            int replies, clicks, pid;
84:            try {
85:               String sql = "select * from posts " + searchStr
```

```
                + " order by plastUpdate desc limit " +
                  (pageno - 1) * pagesize + "," + pagesize;
86：                    rs = st.executeQuery(sql);
87：              //后面的代码同前一小节，略
88：             %>
89：             </div>
90：             </div><!-- end of postlist-->
91：          </div> <!--end of content-->
92：           <%@include file="/foot.jsp" %>
93：       </div>
94：</body>
```

代码说明如下。

- 代码第 3～8 行用以生成搜索的条件子句。
- 第 10～17 行提取页号参数的值，表示要显示的是哪页的记录，若该参数值为空，则取默认值 1。
- 第 18～34 行创建了数据库连接及用以执行 SQL 语句的 Statement 对象 st。
- 第 36～41 行计算出符合条件的记录总数，这里通过执行查询语句“select count(*) from posts”的形式获取记录总数。
- 第 42～47 行以当前页号为中点，计算前后各 3 页的页号。
- 第 59～76 行代码显示了页号链接，显示在“新帖”链接之后，搜索帖子的表单之前。
- 第 85 行的查询语句可根据页号变量 pageno 的值，查询出一页的记录。
- 其余代码与前面相同，不再一一说明。

8.3.4　重点提示

- 要对查询结果进行分页，只需构造适当的 SQL 语句，使之只查询出指定页的结果集即可。
- 分页时还应根据实际情况，提供适当的页号导航链接，以便实现翻页的功能。
- 若要使分页代码具有更强的通用性，可将每页显示的记录条数设置为应用的参数，或在属性文件中定义。



思考与训练

一、思考题

1．可通过哪些方法实现数据的分页显示？请比较这些方法的优劣。

2．了解目前有哪些主流的分页显示组件，试掌握其中一种的使用方法。

二、综合实训

1．在数据库中创建学生成绩表，显示该表的所有记录，并在页面上提供条件搜索，可根据姓名关键字搜索记录。

2．为上一题的学生成绩显示页面添加分页显示功能。

3．请编写帖子详情页面，以显示某个帖子的标题、内容、作者信息，以及相关的回复帖的标题、作者、内容。

第 9 章　回帖功能的实现

9.1　任 务 提 出

论坛是访问者各抒己见的一个平台，用户除了可以发表自己的主题帖外，在查看一个帖子的内容和回复时，还应该可以针对该主题帖发表自己的意见，也就是平时所称的“发表回复”功能。

9.2　输入回复帖的页面

由于输入的回复内容可能较长，一般可通过 HTML 的多行文本框标签<textarea>来提供输入，若需要提供较为丰富的编辑功能，则可选择在线编辑器来实现。本章将学习多行文本框与在线编辑器的使用。

9.2.1　HTML 标签——textarea

HTML 中的 textarea 标签可提供页面上的一个多行文本框，其格式为：

```
<textarea name="" cols="" rows="">初始内容</textarea>
```

name 表示该多行文本框的名称，而 cols 和 rows 属性分别表示显示区域的列数和行数，若实际输入的内容超出了这个限制，则通常会出现水平或垂直方向的滚动条。在标签<textarea>和</textarea>之间可以填写文字，它们将作为文本框的初始内容显示在文本框中。

【例 9.1】　多行文本框的使用。

主要代码　web\9\9_1\textarea.html

```
1：<body>
2：    <form   action="save.jsp">
3：      <textarea name="content" cols="60" rows="10">在此填写意见
4：      </textarea>
5：      <br/>
6：      <input type="submit"     value="确定"/>
7：      </form>
8：</body>
```

该例的效果如图 9-1 所示。

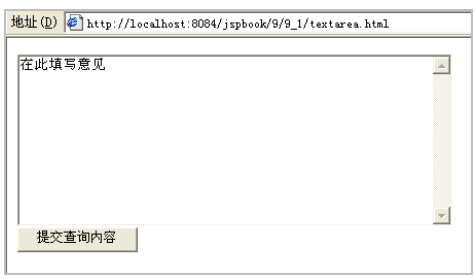


图 9-1　多行文本框的使用效果

9.2.2　在线编辑器 FCKeditor 的使用

我们通过 HTML 的 textarea 标签，可以输入较长的文本内容，但是所输入的内容是纯文本的格式，比较单一，无法控制内容的字体、字号、颜色等。那么，是否能像微软的 Office 软件中的 Word 那样，在输入内容时提供丰富的编辑功能呢？通过编写 JavaScript 程序可以实现这样的在线编辑功能。不过，由于这样的需求比较普遍，因此目前已有多款的在线编辑器软件可供使用，FCKeditor 是其中非常著名的一款。

FCKeditor 是一款功能强大的免费软件，可以通过网站下载其最新版本，下载地址为 http://www.fckeditor.net/download 或 http://ckeditor.com/download。这里以 2.6 版为例说明其使用方法。

从官方网站上下载 FCKeditor 压缩包，将得到的文件 FCKeditor_2.6.3.zip 解压，得到 fckeditor 目录，按照以下步骤在项目里添加 FCKeditor，并在页面中使用。

（1）将 fckeditor 目录的内容只保留如图 9-2 所示的 4 个文件和 1 个文件夹，然后把 fckeditor 目录复制到 web 文件夹下，即和 JSP 页面文件处于同一个目录。

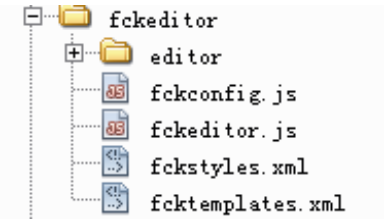


图 9-2　FCKeditor 的主要文件

其中的 fckeditor.js 是其核心文件，fckconfig.js 则是配置文件。

（2）若页面文件直接存放在 web 目录下，则可在需要的页面里添加以下代码。

在<head></head>间添加对.js 文件的引用：

```
<script type="text/JavaScript" src="fckeditor/fckeditor.js"></script>
```

在<body></body>间需要显示在线编辑器的地方添加如下代码：

```
<script type="text/JavaScript">
    var sBasePath="fckeditor/";
    var oFCKeditor = new FCKeditor('pcontent') ;
    oFCKeditor.BasePath = sBasePath ;
    oFCKeditor.Height = 400 ;
    oFCKeditor.Value = '' ;
    oFCKeditor.Create() ;
</script>
```

代码“var oFCKeditor = new FCKeditor('pcontent')”表示输入框的名称是 pcontent，当数据提交到指定页后，可在接收页面中使用“request.getParameter("pcontent")”获取它的值。

FCKeditor 在线编辑器的界面如图 9-3 所示。

可以看到它的编辑功能非常强大，几乎可以与 Word 媲美。不过，实际使用时往往不需要这么复杂的功能。通过修改 fckeditor 目录下的配置文件 fckconfig.js，可以定制自己的工具条。

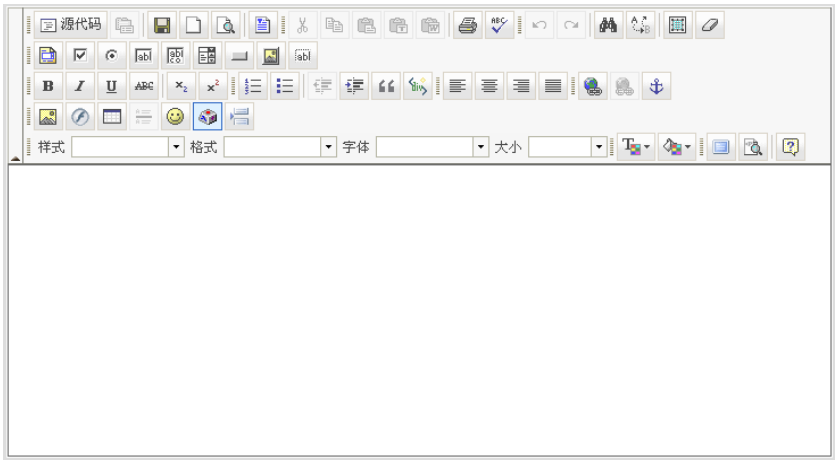


图 9-3　FCKeditor 的默认运行效果

打开此文件，找到其中的 FCKConfig.ToolbarSets 配置段，这部分代码如下：

```
1：FCKConfig.ToolbarSets["Default"] = [
2：['Source','DocProps','-','Save','NewPage','Preview','-','Templates'],
3：...
4：] ;
5：
6：FCKConfig.ToolbarSets["Basic"] = [
7：        ['Bold','Italic','-','OrderedList','UnorderedList','-','Link',
   'Unlink','-','About']
8：] ;
9：
10：FCKConfig.ToolbarSets["Custom"] = [
11：        ['Source'],['Cut','Copy','Paste','PasteText','PasteWord','-',
  'SpellCheck'],
12：        ...
13：] ;
```

这里分别配置了 3 种不同形式的工具栏。“FCKConfig.ToolbarSets["Default"]”表示默认工具栏的配置；“FCKConfig.ToolbarSets["Basic"]”对应于只具备基本功能的工具栏；“FCKConfig.ToolbarSets["Custom"]”表示自定义工具栏的配置。一般来说，我们可以通过修改“FCKConfig.ToolbarSets["Custom"]”数组的内容，来定义自己的工具栏。该数组的原始设置是与 Default 默认工具栏相同的。数组中的字符串即表示工具栏上的一个按钮，特殊字符“/”表示换行。

若要使生成的在线编辑器使用自定义的工具栏，则应在生成 FCKeditor 的一个对象时，添加工具栏的设置，代码如下：

```
oFCKeditor.ToolbarSet="Custom";
```

一个自定义的工具栏效果如图 9-4 所示。



图 9-4　自定义的 FCKeditor 工具栏

在页面中引入对 fckeditor.js 文件的引用时，若页面文件并非直接存放在项目的 web 目录下，而是在 web 的子目录中时，需要注意指明被引用文件的路径。具体写法可参考 jspbook 项目的 web\9\9_2 中 testFck.jsp 文件的代码。

若要使用 FCKeditor 提供的文件上传功能，则需要再下载相应的 Java 压缩包，并在项目中进行相应的配置。

这部分内容请浏览官方网站的相关文档，这里不做详细的介绍。

9.2.3　任务实现

当用户在浏览帖子的内容及回复时，可以在帖子详情页面通过单击“回复”链接转向发表回复的页面，帖子详情页面的效果如图 9-5 所示。

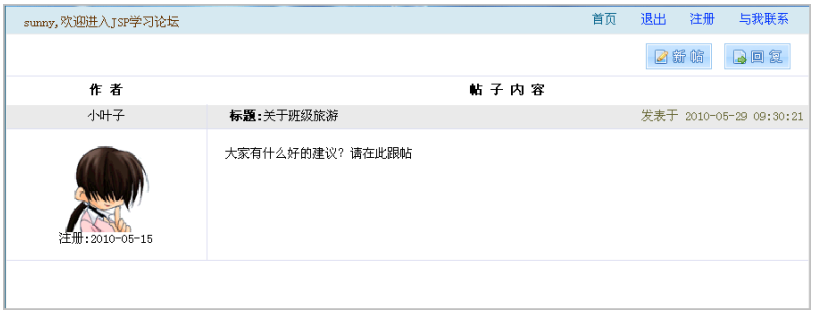


图 9-5　帖子详情页面

要完成帖子回复的功能，需要编写以下两个页面。

- 输入回复的页面 newReply.jsp，该页面使用多行文本框或在线编辑器 FCKeditor 提供输入回复内容的功能。
- 保存回复内容的处理页面 saveReply.jsp，当用户回复的内容不少于指定的字数时，将用户输入的回复标题及内容保存到回复帖表 reply 中。

回复内容输入页面 newReply.jsp 的代码如下所示，该页面使用在线编辑器输入回复内容：

程序清单　文件 web\9\9_2\newReply.jsp

```
1：<%@page contentType="text/html" pageEncoding="UTF-8"%>
2：<%
3：        int pid = 0;
4：        if (request.getParameter("pid") != null) {
5：            try {
6：                pid = Integer.parseInt(request.getParameter("pid"));
7：            } catch (Exception e) {
8：                e.printStackTrace(System.err);
9：            }
```

```
        }
        if (pid == 0) {
            out.println("<script>history.back();</script>");
            return;
        }
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP 学习论坛--发表回复</title>
        <link href="css/common.css" rel="stylesheet" type="text/css" />
        <script src="js/common.js"></script>
        <script type="text/JavaScript"
  src="<%=request.getContextPath()%>/fckeditor/fckeditor.js">
        </script>
        <script>
            function postChk()
            {
                if ($("ptitle").value==null||$("ptitle").value=="")
                {
                    $("ptitle").value="回复:";
                }
                var oEditor = FCKeditorAPI.GetInstance("pcontent");
                var content = oEditor.GetXHTML(true)
                content = content.replace(/<[^<]+>/ig, "");
                if (content.sLength()<4)
                {
                    alert("内容太少啦.");
                    return false;
                }
                else
                    return true;
            }
        </script>
    </head>
<body onload="$('ptitle').focus()">
  <div id="container">
      <%@include file="head.jsp" %>
      <div id="content">
          <form action="saveReply.jsp" method="post"
       onsubmit="return postChk()">
              <input type="hidden" name="pid" value="<%=pid%>"/>
              <table style="position:relative">
                  <caption>请输入回复</caption>
```

```
54:                 <tr><th>标题</th>
55:    <td>
56:       <input type="text" name="ptitle" id="ptitle" value="" size="90"/>
57:        </td></tr>
58:       <tr><td colspan="2">
59:            <script>
60:               var sBasePath="fckeditor/";
61:               var oFCKeditor = new FCKeditor( 'pcontent' ) ;
62:               oFCKeditor.BasePath  = sBasePath ;
63:               oFCKeditor.Height     = 300 ;
64:               oFCKeditor.ToolbarSet="Custom";
65:               oFCKeditor.Width=700;
66:               oFCKeditor.Value      = '' ;
67:               oFCKeditor.Create() ;
68:            </script>
69:          </td></tr>
70:       <tr>
71:       <td colspan="2" style="text-align:center">
72:       <input type="submit" value="发表" /> 
73:       <input type="button" value="放弃" onclick="history.back()"/>
74:       </td>
75:       </tr>
76:       </table>
77:      </form>
78:  </div> <!--end of content-->
79:  <%@include file="foot.jsp" %>
80:  </div> <!--end of container-->
81:  </body>
82: </html>
```

代码说明如下。

- 第 2～15 行代码判断请求参数中是否有相关主题帖的 id，如果没有，则无法继续回复操作，因此通过第 12、13 行代码退回到前一页面，并结束程序。
- 第 24～25 行引入使用 FCKeditor 所需要的.js 文件。
- 第 26～44 行自定义了一个 JS 函数 postChk，判断所输入的内容的长度是否达到要求。
- 第 50～77 行提供了一个输入表单，包含 3 项数据：隐藏域 pid，表示主题帖的 id；输入标题的单行文本框 ptitle；第 59～68 行通过 FCKeditor 创建了一个在线编辑框，名称为 pcontent，用以输入回复的内容。
- 第 50 行的 form 标签表示发生提交事件时，将调用前面编写的 JS 函数 postChk 检测所输入的标题和内容的长度，若符合要求，则将表单数据提交到 savePost.jsp 页面进行后续的处理。

当用户填写了回复帖的标题和内容后，将发送给 saveReply.jsp 页面进行处理。该页面接收了来自客户端的数据，而后将其插入到回复帖表 reply 中，具体代码如下：

主要代码　文件　web\9\9_2\saveReply.jsp

```
1：<%@page contentType="text/html" pageEncoding="UTF-8"
 import="java.util.Date,java.sql.*,java.text.*"%>
2：//此处略去部分 HTML 标签
3：<body>
4：    <%
5：    request.setCharacterEncoding("UTF-8");
6：    String ptitle, pcontent, pauthor, pid;
7：    pauthor = (String) session.getAttribute("username");
8：    ptitle = request.getParameter("ptitle");
9：    pcontent = request.getParameter("pcontent");
10：    pid = request.getParameter("pid");
11：    if (ptitle == null) {
12：        ptitle = "";
13：    }
14：    if (pcontent == null || pcontent.length() < 2 || pid == null) {
15：        out.print("<script>alert('内容太少啦.');</script>");
16：        response.sendRedirect("newReply.jsp?pid=" +
        request.getParameter("pid"));
17：        return;
18：    }
19：    String driverClass = "org.gjt.mm.mysql.Driver";
20：    String url = "jdbc:mysql://localhost:3306/bbs
                    ?useUnicode=true&characterEncoding=utf8";
21：    String username = "root";
22：    String password = "rootroot";
23：    Connection con = null;
24：    PreparedStatement pt = null;
25：    try {
26：        Class.forName(driverClass);
27：        con = DriverManager.getConnection(url, username, password);
28：        String sql = "insert into reply(rtitle,rcontent,rauthor,pid)
                        values(?,?,?,?)";
29：        pt = con.prepareStatement(sql);
30：        pt.setString(1, ptitle);
31：        pt.setString(2, pcontent);
32：        pt.setString(3, pauthor);
33：        pt.setInt(4, Integer.parseInt(pid));
34：        pt.executeUpdate();
35：        SimpleDateFormat sm = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
36：        Date now = new Date();
37：        pt.executeUpdate("update posts set preplies=preplies+1,plastUpdate='"
     + sm.format(now) + "',plastAuthor='" + pauthor + "' where pid=" + pid);
    //回复数加 1，最后回复时间更新
38：        out.print("<script>alert('回复成功.');</script>");
39：        response.sendRedirect("postDetail.jsp?pid=" + pid);
40：    } catch (Exception e) {
```

```
41:         e.printStackTrace(System.err);
42:     } finally {
43:         try {
44:             pt.close();
45:         } catch (Exception e) {
46:             e.printStackTrace(System.err);
47:         }
48:         try {
49:             con.close();
50:         } catch (Exception e) {
51:             e.printStackTrace(System.err);
52:         }
53:     }
54:  %>
55: </body>
```

代码说明如下。

- 第 1 行的 page 指令设置的 import 属性导入了所需要的包或类。java.sql 是访问数据库所需要的包，java.util.Date 类用以取得当前的时间，java.text 包中的 SimpleDateFormat 类可将当前时间转换为对应的格式化字符串。
- 第 7 行从 session 中获取当前登录用户的用户名，作为发帖的作者。
- 第 8～10 行从请求对象 request 中提取了用户提交的回复帖的标题、内容和 id。
- 第 11～18 行检测所输入的内容是否太短，若太短，则在给出提示信息后，重定向到输入回复的页面 newReply.jsp，程序结束。
- 第 19～22 行定义了 4 个变量，用以表示连接 MySQL 数据库 bbs 所需要的参数。
- 第 23 和 24 行定义了连接对象 con 和预编译语句对象 pt。
- 第 25～54 行完成了记录写入的工作。其中第 26 行加载了驱动类，第 27 行创建了到指定数据库的连接，第 28 行构造了将数据插入表 reply 中的 SQL 语句，其中有 4 个占位符，由 4 个“?”表示，第 29 行从连接对象 con 中创建一个预编译对象并将 SQL 语句装入其中，第 30～33 行设置了对应于 4 个占位符的参数值，第 34 行执行该 SQL 语句。
- 除了将记录值写入回复帖表中，还应修改主题帖表中相应记录的回复数、最后作者及最后发表时间，第 35～37 行完成了该项工作。

页面运行效果如图 9-6 所示。



图 9-6　输入回复的标题和内容

回复成功后，将跳转到帖子详情页面，此时可以看到回复的内容已经出现在页面中了，如图 9-7 所示。



图 9-7　回复成功

9.2.4　重点提示

- 在页面中使用 FCKeditor 时，首先需在页面中添加对 fckeditor.js 文件的引用。
- 可以通过修改配置文件 fckconfig.js，定制自己的工具栏。
- 创建在线编辑器的方式有两种：一种是使用 Create 方法直接创建；另一种是先在页面上创建一个多行文本框，而后使用 FCKeditor 的 ReplaceTextarea 方法以同名的 FCKeditor 对象替换掉该多行文本框。



思考与训练

一、思考题

1．若希望能根据用户的选择提供多行文本框或在线编辑器以输入文本内容，应如何实现？

2．请阅读 FCKeditor 的相关文档，了解其上传功能的使用方法。

3．目前还有哪些较流行的在线编辑器组件？请查找相关资料。

二、综合实训

1．请结合在线编辑器组件完成论坛的发表新帖功能。

2．用户在浏览帖子时，若某个主题帖或回复帖是自己发表的，则可以对该帖子的内容进行编辑，请使用在线编辑器组件完成帖子的编辑功能。

第 10 章　功能的增强与改进

本章将改进前面完成的注册功能，在改进或添加功能的过程中进一步学习 JavaBean、Servlet、文件上传及 Ajax 等知识。

10.1　数据库通用处理类

10.1.1　任务提出

前面的章节中，在涉及到数据库访问的处理时，采用了直接在页面文件中编写连接代码的形式，这些代码被一次次地重复编写。这种方式除了增加编程的代码量外，还给系统的维护带来了极大的不便。例如，日后需要修改连接的参数等内容时，需要将所有页面中的此类代码进行修改，工作量之大可想而知。那么，是否有其他更为高效更便于维护的方式呢？经过分析，可以知道，访问数据库时主要包括以下几种处理：

- 创建数据库连接；
- 执行查询语句；
- 执行增删改或数据定义（DDL）语句；
- 关闭各项数据库资源。

若将这些处理过程封装到一个 Java 类中，则在需要时可直接调用该类完成相应的处理。为了进一步提高代码的通用性，降低代码与使用环境的关联程度，可将连接数据库所需要的参数单独存储在一个属性文件中，连接数据库时从此文件中读取即可。这样，即使连接数据库的参数发生了改变，也不需要重新修改代码。

10.1.2　编写通用处理类

创建数据库连接时，需要指明连接数据库的驱动类名称、连接 URL、用户名及密码。我们将这些参数保存在属性文件 dbconn.properties 中；在 NetBeans 中，该文件可存放在项目的 src\java 目录下；在 MyEclipse 中，可存放在项目的 src 目录下。文件内容如下：

```
文件内容 src\java\dbconn.properties
driver = org.gjt.mm.mysql.Driver
url = jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=utf8
username = root
password = rootroot
```

这里连接的是 MySQL 数据库，数据库名称为 test，连接用户名是 root，密码为 rootroot，实际使用时应根据具体情况做相应的更改。

通用处理类 Conn.java 的代码如下所示：

```
程序清单　src\java\db\Conn.java
1：package db;
2：
```

```
3：import java.io.InputStream;
4：import java.sql.*;
5：import java.util.Properties;
6：
7：public class Conn {
8：        private Connection con;
9：        private Statement st;
10：//构造函数，在创建 Conn 类的对象时自动调用
11：        public Conn() throws Exception {
12：                Properties pros = new Properties();
13：                InputStream in;
14：                in= Conn.class.getResourceAsStream("/dbconn.properties");
15：                pros.load(in);
16：                in.close();
17：                String driver = pros.getProperty("driver");
18：                String url = pros.getProperty("url");
19：                String username = pros.getProperty("username");
20：                String password = pros.getProperty("password");
21：                Class.forName(driver);
22：                con = DriverManager.getConnection(url,username,password);
23：                st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                   ResultSet.CONCUR_UPDATABLE);
24：        }
25：//executeQuery--执行查询语句
26：        public ResultSet executeQuery(String sql) {
27：                ResultSet rs = null;
28：                try {
29：                        rs = st.executeQuery(sql);
30：                } catch (Exception e) {
31：                        e.printStackTrace(System.err);
32：                }
33：                return rs;
34：        }
35：  //executeUpdate 方法--执行对数据库的更新的 SQL 语句
36：        public int executeUpdate(String sql) {
37：                try {
38：                        return st.executeUpdate(sql);
39：                } catch (Exception e) {
40：                          e.printStackTrace(System.err);
41：                        return 0;
42：                }
43：        }
44：//创建预编译对象的方法
45：        public PreparedStatement prepareStatement(String sql) {
46：                try {
47：                        return con.prepareStatement(sql);
```

```
48:             } catch (SQLException e) {
49:                 e.printStackTrace(System.err);
50:                 return null;
51:             }
52:         }
53: //关闭各项资源的方法
54:         public void close() {
55:             try {
56:                 if (st!=null) st.close();
57:             } catch (Exception e) {
58:                 e.printStackTrace(System.err);
59:             }
60:             try {
61:                 if (con!=null) con.close();
62:             } catch (Exception e) {
63:                 e.printStackTrace(System.err);
64:             }
65:         }
66: }
```

代码说明如下。

- 该类共有 5 个方法。构造方法 Conn，执行 select 语句的方法 executeQuery，执行 insert、delete、update 等 SQL 语句的方法 executeUpdate，返回一个预编译的 PreparedStatement 对象的方法 prepareStatement，以及逆序关闭各项资源的方法 close。
- 在构造方法中通过第 12～20 行读取了存放在 src\java 目录下面的属性文件 dbconn.properties 中的 4 个属性，第 21 行加载了驱动类，第 22 行创建了数据库连接，第 23 行创建了一个 Statement 对象。

10.1.3　使用通用处理类

编写了通用处理类后，其他页面均可通过调用该类来完成数据库的访问。这里以前面第 6 章中已经完成的注册功能为例，说明如何在程序中使用数据库通用处理类。

修改原 6.6 节中的 chkReg.jsp 页面，在 page 指令中使用 impor 属性添加对 db.Conn 类的导入，而后将其中“if (msg.equals(""))｛...｝”部分的代码修改为：

部分代码　web\10\10_1\chkReg.jsp

```
1:    if (msg.equals("")) {
2:        Conn conn = null;//声明连接对象
3:        ResultSet rs = null;     // 声明结果集对象
4:        try {
5:            conn = new Conn(); //获取连接
6:            String sql = "select * from users where uname='" + uname + "'";
7:            rs = conn.executeQuery(sql);
8:            if (rs != null) {
9:                if (rs.next()) {
10:                   msg = "用户名<b><font color='red'> " + uname +
```

```
                " </font></b>已被使用，请改用其他名字.";
11：            }
12：          }
13：          if (msg.equals("")) {//用户名可用，执行插入操作
14：            sql = "insert into users(uname,upw,usex,uemail,ulogo)
                  values('" + uname + "'";
15：            sql += ",'" + upw + "','" + usex + "','" + uemail + "','" +
          ulogo + "')";
16：            int t = conn.executeUpdate(sql);
17：            if (t > 0) {
18：              msg = "恭喜！注册成功!";
19：            } else {
20：              msg = "Sorry,注册失败.";
21：            }
22：          }
23：        } catch (Exception e) {
24：          out.println("网站发生异常.");
25：          e.printStackTrace(System.err);
26：        } finally {
27：          try {
28：            if (rs != null) {
29：              rs.close();
30：            }
31：          } catch (Exception e) {
32：          }
33：          try {
34：            if (con != null) {
35：              conn.close();
36：            }
37：          } catch (Exception e) {
38：          }
39：        }
40：    }
```

代码说明如下。

- 第 2 行和第 5 行声明并创建了一个 Conn 类的对象 conn。
- 第 7 行通过 conn 对象调用 Conn 类的 executeQuery 方法直接执行 select 语句，以查询用户信息表中是否存在与指定用户名相同的记录。
- 第 16 行通过 conn 对象调用 Conn 类的 executeUpdate 方法执行 insert 语句，将新用户的注册信息插入到用户信息表中。
- 第 34～36 行通过 conn 对象调用了 Conn 类的 close 方法，该方法的作用是关闭所创建的连接对象和 Statement 对象。
- 应注意，在此段代码中，并没有直接创建 Statement 对象，也没有直接使用该类型的对象去执行 SQL 语句，所有执行 SQL 语句的工作都是交给 Conn 类的相关方法完成的。

10.2　JavaBean 的使用

10.2.1　任务提出

虽然注册信息保存页面 chkReg.jsp 中调用通用类完成了数据库连接，但仍有大量的 Java 处理代码。这些代码主要完成两个方面的工作，一是检测用户名是否重复，二是将验证后的正确数据插入到表中。本节将介绍 JSP 的组件 JavaBean 的相关知识，以便将注册功能中的业务处理代码做进一步的封装。

10.2.2　JavaBean 的基本概念

JavaBean 一般可分为可视化组件与非可视化组件。可视化组件在运行时有图形界面，如按钮、报表组件等；非可视化组件则主要用于封装数据实体、业务处理逻辑、数据库访问操作等。目前 JavaBean 更多地应用在非可视化领域。在 JSP 中使用 JavaBean，可以更好地将表现层与业务层分离开来，即使用 JSP 页面来显示内容，使用 JavaBean 来封装业务处理逻辑，从而增强程序的可读性、可维护性，提高代码的重用性。

JavaBean 需要遵循的设计规范如下。

- 必须有一个无参的构造函数。
- 必须有包名。
- 对属性进行读写的方法应以 setXxx()和 getXxx()的形式命名，“Xxx”是第一个字母转为大写的属性名。

当一个 JavaBean 中没有显式地定义一个无参构造函数时，系统会自动为其添加一个默认的函数体为空的构造函数。

下面的类 Book 就是一个非常简单的 JavaBean：

程序清单　src\java\chap10\Book.java

```
1：package chap10;
2：public class Book {
3：        private String bname;
4：    public String getBname() {
5：        return bname;
6：    }
7：    public void setBname(String bname) {
8：        this.bname = bname;
9：    }
10：}
```

一般来说，除非有特殊的需要，否则在定义一个 Java 类时，其属性的访问权限为私有（private）的，因此通常把对属性进行读写的方法定义为公共（public）的访问权限。

当然，这里的 Book 类只有一个属性，因此不具有实际的使用价值，仅仅是一个符合规范的 JavaBean 的样例。在实际的应用中，往往会根据所要处理的数据库表结构来定义对应的实体类。此时，实体类的一个实例与表中的一条记录有对应的关系，这种对应关系被称为“对象-关系映

射”，简称 ORM（Object- Relational Mapping），使用 JavaEE 的轻量级框架 Hibernate 可以方便地实现这种映射。

10.2.3　通过 JSP 动作使用 JavaBean

JSP 中提供了一系列的动作来创建或引用 JavaBean 的实例、设置或读取该实例的属性。

<jsp:useBean>动作用于在页面中创建或引用一个 JavaBean 实例，<jsp:setProperty>动作和<jsp:getProperty>动作则用于设置/读取 Bean 实例的属性值。

1．<jsp:useBean>动作

<jsp:useBean>动作的语法格式为：

```
<jsp:useBean class="package.class"   id="id"
                scope="page|request|session|application"/>
```

或

```
<jsp:useBean class="package.class"   id="id"
                    scope="page|request|session|application">
</jsp:useBean>
```

该动作主要指明所要创建或引用的 JavaBean 实例的包名、类名、id 和有效范围等。

该动作的属性说明如下。

- class：指出 JavaBean 类包括包名在内的完整类名。
- id：JavaBean 实例的名称。
- scope：该 JavaBean 实例的有效范围，可以有 4 种取值，分别为 page、request、session、application，默认值为 page。下面对这 4 个取值的含义做具体的说明。
 - page：该 JavaBean 实例仅在本页面范围内有效。
 - request：该 JavaBean 实例在请求所延续的范围内有效。例如，在页面 a.jsp 中使用动作“<jsp:useBean class="sunny.Users" id="u1" scope="request"/>”创建了 request 范围内有效的 JavaBean 实例 u1，若在此页面中使用<jsp:include>动作包含页面 b.jsp，或使用<jsp:forward>动作将请求转发到页面 b.jsp，则在 b.jsp 中的动作“<jsp:useBean class="sunny.Users" id="u1" scope="request"/>”将引用在 a.jsp 页面创建的同一个 JavaBean 实例 u1。
 - session：该 JavaBean 实例在会话范围内有效，即可以在属于同一次会话的页面中使用。更通俗地说，是可以被同一个客户端在该应用的所有页面中使用。
 - application：该 JavaBean 实例可在整个应用范围内使用，即可以被所有客户端在本应用的任意页面中共享使用。

后面章节中将结合<jsp:setProperty>和<jsp:getProperty>给出使用案例。

2．<jsp:setProperty>动作

设置某个已创建的 JavaBean 实例的属性值，该动作实际上是调用 JavaBean 中的 set 方法来实现的。因此，应保证所设置的属性具有 set 方法。该动作可以有以下 3 种形式。

1）直接指定属性的值

```
<jsp:setProperty   name="beanName"   property="propertyName"
                                        value="propertyValue" />
```

指定 JavaBean 实例的某个属性值，该实例的名称由 beanName 指定，属性名称由 propertyName 指定，属性值由 propertyValue 指定。例如：

```
<jsp:setProperty name="u1" property="uname" value="Karen" />
```

或

```
<jsp:setProperty name="u1" property="uname"
                    value="<%=request.getParameter(\"username\")%>"  />
```

2）由请求参数定属性值

```
<jsp:setProperty  name="beanName"  property="propertyName"
                    param="parameterName" />
```

指定 JavaBean 实例的某个属性值，该实例的名称由 beanName 指定，属性名称由 propertyName 指定，属性值为请求参数 parameterName 的值。

例如：

```
<jsp:setProperty name="u1" property="uname" param="username" />
```

3）同名请求参数自动传递值

```
<jsp:setProperty  name="beanName"  property="*" />
```

指定 JavaBean 实例的属性值，若名称为 beanName 的 Bean 实例，其属性名称与请求参数的名称相同，则自动将其值设为同名的请求参数的值。

例如：

```
<jsp:setProperty name="u1"  property="*"  />
```

第 3 种形式可以方便地将请求参数的值传递到 JavaBean 实例中，大家应该尽量掌握这种形式。但应注意，所传递的请求参数的名称应与所定义的 JavaBean 中的属性同名，且 JavaBean 中应有相应属性的 set 方法，否则不能实现数据的自动传递。

3．<jsp:getProperty>动作

返回某个已创建的 JavaBean 实例的某个属性的值。

语法格式：

```
<jsp:getProperty name="beanName" property="propertyName" />
```

例如：

```
<jsp:getProperty name="u1" property="uname" />
```

该代码取出 JavaBean 实例 u1 的属性 uname 的值并输出，使用前应保证实例 u1 已经创建。

10.2.4　JavaBean 的使用案例

下面通过一个简单的案例，演示如何利用 NetBeans 工具自动生成 JavaBean 的 set/get 方法。

【例 10.1】　JavaBean 的简单使用。

在页面 input.jsp 中输入用户名和年龄后，提交到 testBean.jsp，在此页面中使用 JavaBean 封装客户端提交的表单数据，并在页面中显示。

1．创建类文件 Userinfo.java

该 JavaBean 封装了用户名和年龄两个属性及对应的读写方法。

（1）首先，在项目窗口中将光标定位在“源包”上，然后在 NetBeans 的“文件”菜单中选择“新建文件”，或单击工具栏左上角的第一个按钮。

在弹出的文件类型选择窗口中选择“Java”→“Java 类”。

（2）在弹出的对话窗口中输入“类名”Userinfo，“包”名为 chap10，然后单击完成，如图 10-1 所示。

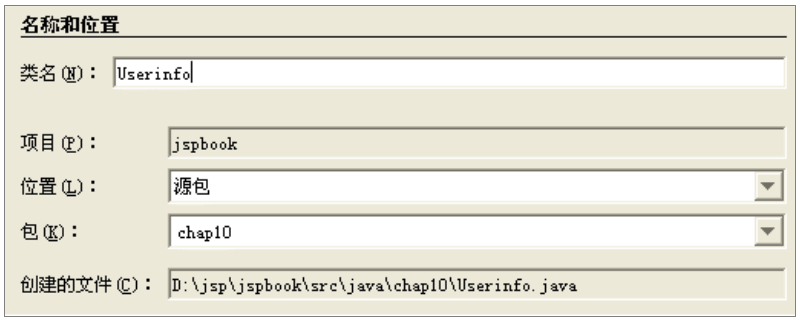


图 10-1　输入新建类的类名与包名

（3）NetBeans 将自动在编辑窗口中打开所创建的 java 文件，可以看到已经按照模板建好了文件的基本结构，如图 10-2 所示。

```
1  /**
2   *   Document    : Userinfo.java
3   *   Created on : 2010-6-8, 23:38:49
4   *   Author      : xuwz
5   */
6  package chap10;
7
8  public class Userinfo {
9  }
```

图 10-2　按照模板生成的 Java 类

（4）我们为该类添加两个属性 name 和 age，表示用户的用户名和年龄，之后把光标定位在如图 10-3 所示的编辑窗口中第 10 行的位置，单击鼠标右键，弹出如图 10-3 所示的右键菜单。

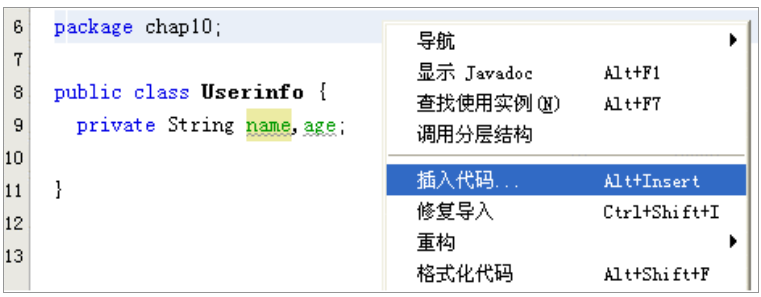


图 10-3　右键菜单

（5）在右键菜单中选择“插入代码”，系统将弹出如图 10-4 左图所示的菜单选项，选择“getter 和 setter”，在下一步的对话窗口中选择需要生成 getter 方法和 setter 方法的属性名，单击下方的“生成”按钮。



图 10-4　生成 setter 和 getter

也可通过主菜单上的“重构”→“封装字段”，打开如图 10-5 所示的对话窗口进行设置。

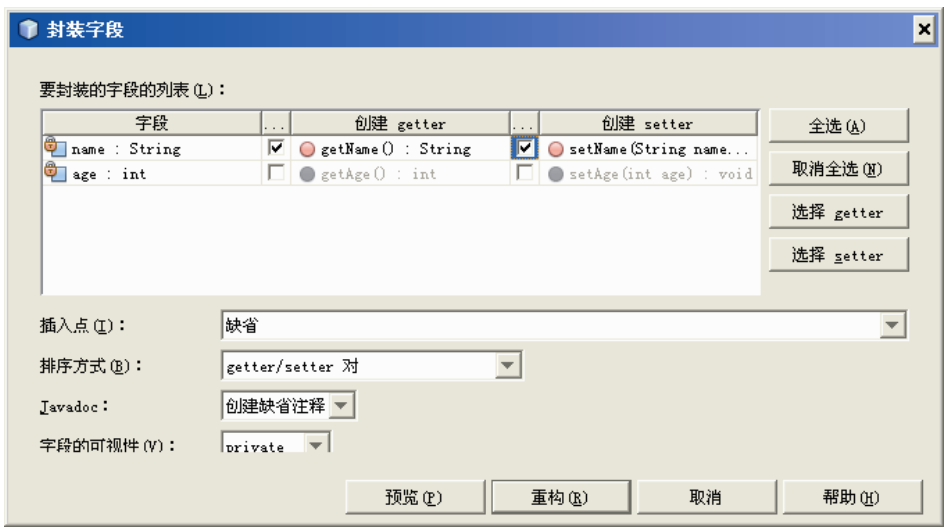


图 10-5　封装字段对话窗口

通过这个对话窗口，可以自由选择所要生成的属性及其读写方法。当我们选择为所有的字段生成 setter 和 getter 时，可得到如下的代码：

程序清单　src\java\chap10\Userinfo.java

```
1：package chap10;
2：public class Userinfo {
3：    private String name;
4：    private int age;
5：
6：    public int getAge() {
7：        return age;
8：    }
9：
```

```
10:    public void setAge(int age) {
11:        this.age = age;
12:    }
13:
14:    public String getName() {
15:        return name;
16:    }
17:
18:    public void setName(String name) {
19:        this.name = name;
20:    }
21: }
```

2．新建页面文件 input.jsp，在其中输入用户名和年龄

主要代码　　文件 web\10\10_2\input.jsp：

```
1:      <body>
2:          <form action="testBean.jsp" method="POST">
3:              用户名<input type="text" name="uname" /><br/>
4:              年龄<input type="text" name="uage" /><br/>
5:              <input type="submit" value="登录" />
6:              <input type="reset" value="重置" />
7:          </form>
8:      </body>
```

创建这个文件时可选择使用 3.2 节中创建的带表单的 JSP 页面模板，或者使用 NetBeans 中的组件面板进行拖放式设计。

3．新建文件 testBean.jsp，在其中创建并使用一个 Userinfo 的实例

主要代码　　文件 web\10\10_2\testBean.jsp

```
1:      <body>
2:          <% request.setCharacterEncoding("utf-8");%>
3:          <jsp:useBean class="chap10.Userinfo" id="u1" />
4:          <jsp:setProperty name="u1" property="*" />
5:          用户名：
6:          <jsp:getProperty name="u1" property="uname"/>
7:          年龄：
8:          <jsp:getProperty name="u1" property="uage" />
9:      </body>
```

代码说明如下。

- 第 3 行创建了 JavaBean 对象 u1。
- 第 4 行自动提取了用户填写的两项数据，并封装在对象 u1 中。
- 第 6 行和第 8 行使用 JSP 动作输出了 u1 的两个属性值。

访问 input.jsp，输入用户名和年龄并提交，案例的运行效果如图 10-6 所示。



图 10-6　案例的运行效果

10.2.5　任务实现

下面，我们使用 JavaBean 完成前面的注册信息处理功能，包括以下两个方面的工作。

- 编写处理用户信息的 JavaBean，该 JavaBean 的属性与用户信息表中各列相对应，用户信息表 users 的结构可参考第 4 章。除各属性的读写方法外，还另包含 3 个方法，以验证必填项是否已填写，用户名是否唯一，并将通过验证的数据写入数据库。
- 在注册信息处理页面调用该 JavaBean 完成注册的处理。

相关代码如下：

主要程序代码　src\java\chap10\Users.java

```
1：package chap10;
2：
3：import db.Conn;
4：import java.sql.PreparedStatement;
5：import java.sql.ResultSet;
6：import java.sql.SQLException;
7：
8：public class Users {
9：    private String uname,upw,upw2,usex,udate,uemail,ulogo;
10：    private int ulevel;
11：
12：public String basicCheck(){
13：      String msg="";
14：if (uname == null || uname.equals("") || upw == null || upw.equals("")){
15：        msg = "请填写用户名和密码.";
16：    } else if (!upw.equals(upw2)) {
17：        msg = "两次输入的密码不一致!";
18：    }
19：      return msg;
20：    }
21：  public String uniqueName(){
22：      Conn conn=null;
23：      ResultSet rs=null;
24：      String msg="";
25：        try {
26：            conn = new Conn();
```

```
27:             String sql="select * from users where uname='"+uname+"'";
28:             rs=conn.executeQuery(sql);
29:             if (rs!=null)
30:                 if (rs.next())
31:                     msg = "用户名<b> " + uname + " </b>已被使用，请改用其他名字.";
32:         } catch (Exception ex) {
33:           ex.printStackTrace(System.err);
34:         }
35:       finally{
36:             try {
37:                 if (rs!=null) rs.close();
38:             } catch (SQLException ex) {
39:                 ex.printStackTrace(System.err);
40:             }
41:           conn.close();
42:       }
43:       return msg;
44:   }
45:
46:   public boolean insert(){
47:       boolean success=false;
48:       Conn conn=null;
49:       PreparedStatement pt=null;
50:         try {
51:             conn = new Conn();
52:                 String sql="insert into users(uname,upw,usex,uemail,ulogo)
          values(?,?,?,?,?)";
53:                 pt = conn.prepareStatement(sql);
54:                 pt.setString(1, uname);
55:                 pt.setString(2, upw);
56:                 pt.setString(3, usex);
57:                 pt.setString(4, uemail);
58:                 pt.setString(5, ulogo);
59:                 success=pt.executeUpdate()>0;
60:         } catch (Exception ex) {
61:           ex.printStackTrace(System.err);
62:         }
63:       finally{
64:             try {
65:                 if (pt!=null) pt.close();
66:             } catch (SQLException ex) {
67:                 ex.printStackTrace(System.err);
68:             }
69:           conn.close();
70:       }
71:       return success;
```

```
72:  }
73:
74:    public String getUname() {
75:       return uname;
76:    }
77:
78:    public void setUname(String uname) {
79:       this.uname = uname;
80:    }
81: //其他属性的 set 与 get 方法，代码略去
82: .....
83: }
```

代码说明如下：

- 与表单的输入项对应地，该 JavaBean 提供了与之对应的各个属性，除变量 upw2 外，其他属性也与用户信息表 users 中的各列一一对应；利用 IDE 工具可自动生成各属性的 set 和 get 方法。限于篇幅，这里除 uname 属性外，其他属性的读写方法均未列出。
- 方法 basicCheck 提供了对必填项及两次输入的密码是否一致等验证的处理，当数据不合法时，原因可能有多种。因此，这里根据不同的原因返回相应的错误信息字符串，而正确时则返回空串。
- 方法 uniqueName 对数据的唯一性进行了检测，用户名不重复时返回空字符串，否则返回相应的提示信息字符串。
- 方法 insert 将数据插入表中，其返回值只需表示插入记录是否成功，因此返回值类型可设为 boolean，当返回值为 true 时，表示插入成功，否则表示插入失败。

这 3 个方法的处理代码与页面 chkReg.jsp 基本相似，这里不再详细说明。

完成 JavaBean 的编写后，可在处理页面 chkReg.jsp 中调用该 JavaBean 完成逻辑处理，主要代码如下所示：

主要代码 web\10\10_2\chkReg.jsp

```
1:   <body>
2:     <%
3:       request.setCharacterEncoding("utf-8");
4:     %>
5:     <jsp:useBean class="chap10.Users" id="newuser" />
6:     <jsp:setProperty property="*" name="newuser" />
7:     <%
8:       String msg=newuser.basicCheck();
9:       if (msg.equals("")) msg=newuser.uniqueName();
10:        if (msg.equals("")){
11:          if (!newuser.insert()) {
12:            msg = "Sorry,注册失败.";
13:          }
14:        }
15:        if (msg.equals("")) {
16:          out.println("恭喜！ 注册成功!");
```

```
17：              response.setHeader("refresh", "3;url=index.jsp");
18：           } else {
19：        %>
20：        <jsp:forward page="reg.jsp">
21：           <jsp:param name="msg" value="<%=msg%>" />
22：        </jsp:forward>
23：        <%}%>
24：    </body>
```

代码说明如下：

- 第 3 行将请求编码设为 UTF-8，以便能正确地读取中文参数。
- 第 5 行使用<jsp:useBean>动作创建了类 chap10.Users 的一个实例，实例名为 newuser。
- 第 6 行使用<jsp:setProperty>动作，按照同名传递的原则，将请求参数（主要为前一页面 reg.jsp 提交的表单参数）的值赋给实例 newuser 的对应属性。
- 第 8 行通过实例 newuser 调用 basicCheck 方法，检测用户是否有输入必填项的内容，两次输入的密码是否一致，返回的结果信息保存在字符串变量 msg 中。
- 若经过 basicCheck 方法验证后，数据合法，则通过第 9 行代码继续调用 uniqueName 方法判断用户名的唯一性。
- 第 10 行判断 uniqueName 方法返回的结果是否为空串，是则表示用户名可以使用，因此可通过第 11 行代码调用 insert 方法完成记录添加工作。
- 第 15～23 行根据结果字符串 msg 的值是否为空串（前面的所有操作都成功完成）跳转到不同的页面。

修改后页面的运行效果与修改前完全相同。

10.2.6　拓展与提高

一般来说，如果与该数据表相关的处理比较少时，可以采取 10.2.5 节的处理方式，即将封装了处理逻辑的方法直接编写在实体 JavaBean 中；而当相关的操作较多时，可分别设计根据表结构映射得到的实体类和封装了与该表相关的数据库处理操作的操作类。例如，前面的注册功能，可分别设计实体类 Users.java 和 操作类 UsersBean.java，它们的结构大致如下：

主要代码　Users.java

```
1：package chap10;
2：public class Users {
3：    private String uname,upw,upw2,usex,udate,uemail,ulogo;
4：    private int ulevel;
5：    public String getUname() {
6：        return uname;
7：    }
8：
9：    public void setUname(String uname) {
10：        this.uname = uname;
11：    }
12：//略去部分代码：其他各属性的 set/get 方法
13：    }
```

主要代码　UsersBean.java

```
1：public class UsersBean{
2：    public String basicCheck(Users u){
3：    // 代码略
4：    }
5：    public String uniqueName(String username){
6：     //代码略
7：    }
8：    public boolean insert(Users u){
9：     //代码略
10：     }
11：}
```

处理页面 chkReg.jsp 中调用 JavaBean 的相关代码也应修改如下：

主要代码 chkReg.jsp

```
1：    <body>
2：      <%
3：        request.setCharacterEncoding("utf-8");
4：      %>
5：      <jsp:useBean class="chap10.Users" id="newuser" />
6：      <jsp:setProperty property="*" name="newuser" />
7：      <%
8：         UsersBean ud=new UsersBean();
9：         String msg = ud.basicCheck(newuser);
10：         if (msg.equals("")) msg=ud.uniqueName(newuser.getUname());
11：         if (msg.equals("")) {
12：            if (!ud.insert(newuser)) {
13：               msg = "Sorry,注册失败.";
14：            }
15：         }
16：//其他代码略，与前面相同
17：    </body>
```

可以看到，这种设计方式使得模块分工更加明确，更好地体现了面向对象的编程思想，代码的结构也更为清晰，建议当处理逻辑比较复杂时采用。若要更好地实现模块间的解耦，建议参考 DAO 设计模式的思想。

10.2.7　重点提示

在使用<jsp:setProperty>动作接收表单数据前，应先加入代码“request.setCharacterEncoding("utf-8");”，以便能正确提取中文的请求参数。

若希望使用“<jsp:setProperty property="*" name="instanceName" />”这种形式为 JavaBean 实例中的属性赋值时，请求参数必须与 JavaBean 实例中的属性同名。

10.3　使用 Servlet 完成用户注册功能

10.3.1　任务提出

在 10.2 节中我们通过 JSP 页面 chkReg.jsp 调用 JavaBean 完成了注册处理，在此页面中，几乎不必向客户端输出显示信息，在这种情况下，可采用特殊的 Java 类 Servlet 来实现相同的功能。

10.3.2　Servlet 简介

在 JSP 技术出现之前，Sun 公司推出了 Servlet 技术来开发 Java Web 应用，此时无论是呈现页面内容还是处理业务逻辑，都采用 Java 语言编程实现，这种开发方式对程序员的要求较高，也在一定程度上阻碍了它的推广。因此，1999 年 6 月 Sun 公司在 Servlet 的基础上，推出了 JSP 技术，而 Servlet 则转为负责业务逻辑的处理及控制。实际上，JSP 页面被客户端第一次请求时，首先就被转化为了一个 Servlet，因此 JSP 页面的本质就是一个 Servlet。本节将学习 Servlet 的基本概念与一般的应用方法。

1．初识 Servlet

Servlet 是运行在 Web 服务器端的 Java 程序，它可以处理来自客户端的请求并发回响应。我们先通过一个简单的例子认识一下它。

【例 10.2】　输出当前时间与欢迎词的 Servlet。

首先编写一个 Servlet 类，一般是 javax.servlet.http.HttpServlet 类的子类。

在 NetBeans 中新建一个文件，然后在类型选择框中选择“Java”→“Java 类”，在“类名”输入框中输入 SimpleServlet，“包”输入框中输入 chap10.p3，然后单击完成。接下来编写类代码，如下所示：

程序清单　文件　src\java\chap10\p3\SimpleServlet.java

```
1：package chap10.p3;
2：
3：import java.io.*;
4：import java.util.Date;
5：import javax.servlet.*;
6：import javax.servlet.http.*;
7：
8：public class SimpleServlet extends HttpServlet {
9：        protected void doGet(HttpServletRequest request,
                                HttpServletResponse response)
10：            throws ServletException, IOException {
11：        response.setContentType("text/html;charset=UTF-8");
12：        PrintWriter out = response.getWriter();
13：        out.println("<html>");
14：        out.println("<head>");
15：        out.println("<title>一个简单的 Servlet</title>");
16：        out.println("</head>");
```

```
17:                 out.println("<body>");
18:                 out.println(new Date());
19:                 out.println("<br/>欢迎来到 Servlet 学习区!");
20:                 out.println("</body>");
21:                 out.println("</html>");
22:                 out.flush();
23:                 out.close();
24:             }
25:             protected void doPost(HttpServletRequest request,
            HttpServletResponse response)
26:         throws ServletException, IOException {
27:             doGet(request, response);
28:         }
29:     }
```

为了能访问上面编写的 Servlet，必须在项目的配置文件 web.xml 中进行配置，该文件位于项目的 web\WEB-INF 目录中。需要在此定义 Servlet 的逻辑名称，并将该名称映射为一个虚拟的访问路径。

在 web.xml 的<web-app>和</web-app>标记间加入以下内容：

```
1:      <servlet>
2:          <servlet-name>simple</servlet-name>
3:          <servlet-class>chap10.p3.SimpleServlet</servlet-class>
4:          <load-on-startup>1</load-on-startup>
5:      </servlet>
6:      <servlet-mapping>
7:          <servlet-name>simple</servlet-name>
8:          <url-pattern>/abcServlet</url-pattern>
9:      </servlet-mapping>
```

在浏览器中输入如下的路径访问该 Servlet，效果如图 10-7 所示。



图 10-7　访问 Servlet 的运行效果

可以看到，浏览器的 URL 中使用了在 web.xml 中所指定的映射路径 /abcServlet 来访问，而浏览器中呈现的则是该 Servlet 类的 doGet 方法所输出的内容。

总结上面的步骤可以得知，编写一个 Servlet 的一般流程是，继承 HttpServlet，重写 doGet 或 doPost 方法，编写相应的配置文件 web.xml 对 servlet 进行访问配置。

那么，当服务器接收到对一个 Servlet 的请求时，是如何进行处理的呢？下面介绍的有关 Servlet 生命周期的知识，会让我们对此有更深入的了解。

2．Servlet 的生命周期

Servlet 是在容器中运行的，其实例的创建及销毁等都不是由程序员决定的，而是由容器进行控制的。从被容器加载、实例化，到最后被销毁这段时间，称为 Servlet 的生命周期。具体地说，一般包括加载、实例化、初始化、处理客户端请求、销毁这样一个过程。

1）*加载与实例化*

一个 Servlet 会在下列时刻被容器加载到内存。

- 若在配置文件中为 Servlet 配置了自动加载项，则在服务器启动时自动载入。
- 若未配置自动加载项，则在服务器启动后，接收到对 Servlet 的首次请求时，将载入该 Servlet。
- 已经加载的 Servlet，在其源文件被更新后，销毁原实例并重新加载。

一个 Servlet 是否在服务器启动时被自动加载，可通过在 web.xml 中设置此 Servlet 的<load-on-startup>属性来指定。

当服务器加载了 Servlet 后，接着就创建了一个该 Servlet 的实例。

2）*初始化*——init *方法*

在 Servlet 实例开始处理任何客户端请求之前，容器调用 Servlet 的 init 方法，对 Servlet 进行初始化。

该方法只在加载后被执行一次，以后直到销毁前都不再被调用。除非有特殊的需要，否则可以不重写这个方法。

3）*请求处理*——service *方法*

当 Servlet 完成初始化后，就一直存在于容器中，可以开始接收并服务于客户端的请求。

当服务器接收到对 Servlet 的请求时，若 Servlet 已被加载且实例化，则调用该 Servlet 的 service 方法，该方法有两个参数 request 和 response，分别为 HttpServletRequest 类型和 HttpServletResponse 类型的对象，参数 request 封装了来自客户端的请求信息，response 则封装了将发往客户端的响应信息。

HttpServlet 的 service 方法会根据客户端请求的不同类型，将请求分发给不同的方法去处理，如将 GET 请求分发给 doGet 方法处理，将 POST 请求分发给 doPost 方法处理。一般来说我们处理的也多数是这两类请求，正如我们在前面例子中看到的一样。因此我们一般编写的是 doGet 或 doPost 方法。

当我们在浏览器地址栏中直接输入 URL，或在页面上通过超链接访问一个 Servlet 时，发出的是 GET 请求。若通过提交表单的方式发起请求，且表单的 method 属性设为 post，则发起的是 POST 请求。

4）*销毁*——destroy *方法*

当 Web 容器决定销毁 Servlet 时，先调用 Servlet 的 destroy 方法。例如，Servlet 的代码被更新时，将调用此方法销毁原来已经加载的实例。当然，当服务器关闭时，也会销毁 Servlet。

3．Servlet 的类继承结构

从上面的例子可以看出，我们编写的 SimpleServlet 是 HttpServlet 类的子类。事实上，与 Servlet 相关的接口和类主要有 3 个，即 javax.servlet.Servlet、 javax.servlet.GenericServlet 和 javax.servlet.http.HttpServlet，它们之间的继承关系如图 10-8 所示。

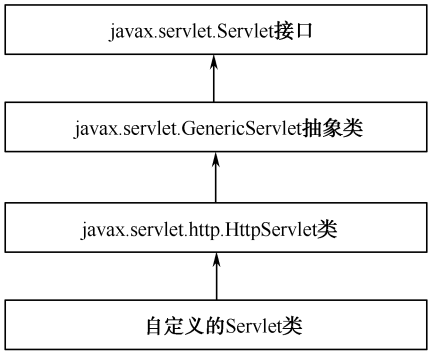


图 10-8　Servlet 的类继承结构

4．JSP 的常用内置对象在 Servlet 中的对应处理

当我们在 JSP 页面中编写代码时，常会使用内置对象 request、response、out 及 session，有时还会用到 application 对象。在 Servlet 中编写代码时，也可能要用到与这些内置对象等同的对象，下面介绍在 Servlet 中取得这些等同对象的方法。

1）request 与 response

在 Servlet 中可以方便地得到对这两个对象的引用，因为 doGet 或 doPost 方法的两个参数 request 和 response 即对应于这两个对象。

2）out

out 对象代表向客户端发送信息的输出流，其类型是 java.io.PrintWriter，在 Servlet 中可通过下面的代码得到与 JSP 中的 out 对象等同的输出流对象：

```
PrintWriter out=response.getWriter();
```

使用前应添加对相应的包或类的导入语句。

3）session

会话对象 session 是 HttpSession 类型的对象，可通过下面的代码得到与某个客户端相关联的会话对象：

```
HttpSession session=request.getSession(true);
```

4）application

application 为 ServletContext 类型的对象，无论有多少客户端在访问，每个应用只有一个 application 对象。在 Servlet 中得到 application 对象的代码如下：

```
ServletContext application=session.getServletContext();
```

10.3.3　使用 Servlet 接收表单数据

当表单向 Servlet 提交数据时，Servlet 中仍可使用 request 对象获取各个请求参数的信息。下面的案例演示了在页面 input.jsp 中输入一项表单数据 msg 后，在 Servlet 中如何提取并显示该信息。

【例 10.3】　Servlet 提取表单数据并显示。

输入页面 inputMsg 将表单数据提交给本应用中访问路径为/showMsgServlet 的 Servlet，代码如下所示：

主要代码 web\10\10_3\inputMsg.jsp

```
1：<body>
2：<form method="POST"
 action="<%=request.getContextPath()%>/showMsgServlet">
3：    <input type="text" name="msg" /><br/>
4：        <input type="submit" value="确定" />
5：        <input type="reset" value="重置" />
6：  </form>
7：</body>
```

第 2 行代码中的表达式 request.getContextPath()表示获取当前应用对应的访问路径，通常为项目的名称。在 NetBeans 中可在项目窗口中右击项目名称，打开“属性”对话窗口，在“运行”选项的“上下文路径”中修改此设置。

下面编写 Servlet 类。前面按编写普通 Java 类的方式完成了 Servlet 的编写，这种方式要求程序员熟练掌握 Servlet 及配置文件的结构。实际上，NetBeans IDE 提供了生成 Servlet 的向导，可以方便地按照模板生成 Serlvet 的基本结构，并自动生成 web.xml 中的配置段。

首先，新建一个文件，文件类型选择“Web”→“Servlet”，如图 10-9 所示。

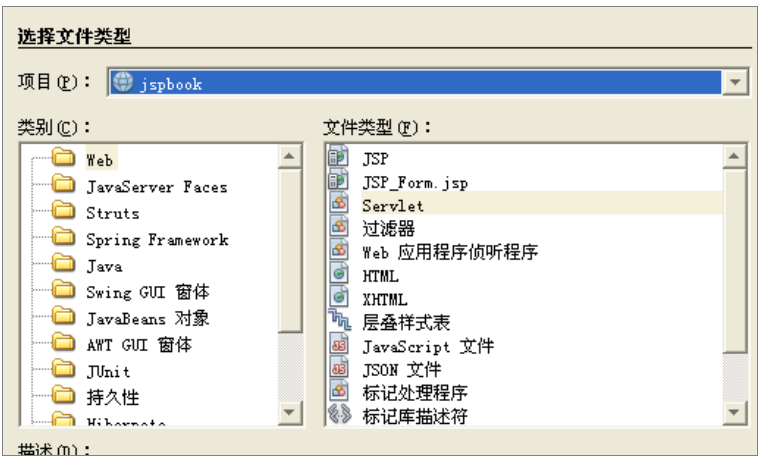


图 10-9　选择 Servlet 文件类型

在随后的对话框中输入类名和包名，单击“下一步”按钮。在如图 10-10 所示的对话窗口中输入 Servlet 的逻辑名称及访问路径，然后单击“完成”按钮，将得到根据 Servlet 的模板自动生成的代码，其中主要包含有以下 4 个方法：

■ protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws Servlet Exception, IOException;

■ protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException;

■ protected void doPost(HttpServletRequest request, HttpServletResponse response)throws ServletException, IOException;

■ public String getServletInfo();



图 10-10　配置 Servlet

在自动生成的代码中，doGet 和 doPost 方法均调用 processRequest 方法，因而此时无论客户端以 GET 方式还是 POST 方式发起请求，都将得到相同的访问结果。getServletInfo 方法返回关于当前 Servlet 的描述信息，一般保留默认代码即可。

修改 processRequest 方法，去掉输出代码段前后的注释，修改后该方法的代码如下所示：

主要代码　showMsgServlet 类的 processRequest 方法

```
1:  protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
2:  throws ServletException, IOException {
3:      response.setContentType("text/html;charset=UTF-8");
4:      PrintWriter out = response.getWriter();
5:      try {
6:          out.println("<html>");
7:          out.println("<head>");
8:          out.println("<title>Servlet ShowMsgServlet</title>");
9:          out.println("</head>");
10:         out.println("<body>");
11:         out.println("<h1>Servlet ShowMsgServlet at " +
                    request.getContextPath () + "</h1>");
12:         out.println("您输入的信息是："+request.getParameter("msg"));
13:         out.println("</body>");
14:         out.println("</html>");
15:     } finally {
16:         out.close();
17:     }
18: }
```

其他代码保持不变，访问 inputMsg.jsp 页面，运行结果如图 10-11 所示。



图 10-11　在 Servlet 中读取请求参数

在本例中，通过表单以 POST 方式向 Servlet 发起了请求，因此实际上是调用的 Servlet 中的 doPost 方法。若在浏览器的地址栏中直接输入网址 http://localhost:8084/jspbook/showMsgServlet?msg=test 进行访问，则将调用 Servlet 中的 doGet 方法。

10.3.4　任务实现

下面修改注册功能的相关代码，改用 Servlet 实现相同的功能。

首先，修改注册信息输入页面 reg.jsp，只需修改 form 标签的 action 属性，如下所示：

```
<form action="<%=request.getContextPath()%>/ChkRegServlet" .../>
```

其次，使用 NetBeans 的向导生成名为 ChkRegServlet 的 Servlet，编写其 doPost 方法，代码如下：

程序清单　src\java\chap10\p3\ChkRegServlet.java

```
1：package chap10.p3;
2：
3：import chap10.p2.Users;
4：import java.io.IOException;
5：import javax.servlet.RequestDispatcher;
6：import javax.servlet.ServletException;
7：import javax.servlet.http.HttpServlet;
8：import javax.servlet.http.HttpServletRequest;
9：import javax.servlet.http.HttpServletResponse;
10：
11：public class ChkRegServlet extends HttpServlet {
12：    protected void doPost(HttpServletRequest request,
                                        HttpServletResponse response)
13：            throws ServletException, IOException {
14：        request.setCharacterEncoding("UTF-8");
15：        Users u = new Users();
16：        u.setUname(request.getParameter("uname"));
17：        u.setUpw(request.getParameter("upw"));
```

```
18:         u.setUpw2(request.getParameter("upw2"));
19:         String msg = u.valid();
20:         if (msg.equals("")) {
21:             u.setUemail(request.getParameter("uemail"));
22:             u.setUlogo(request.getParameter("ulogo"));
23:             u.setUsex(request.getParameter("usex"));
24:             if (!u.insert()) {
25:                 msg = "Sorry,注册失败.";
26:             }
27:         }
28:         request.setAttribute("msg", msg);
29:         RequestDispatcher rd =
             request.getRequestDispatcher("/chkRegResult.jsp");
30:         rd.forward(request, response);
31:     }
32: }
```

代码说明如下。

- 由于在注册输入页面 reg.jsp 中，表单是以 POST 方式提交的，因此，此 Servlet 中只需要提供 doPost 方法。
- 第 15～18 行将用户提交的用户名和两次输入的密码封装到 Users 类的对象 u 中。
- 第 19 行调用 Users 类的 valid 方法，验证用户名是否唯一，以及用户名和密码是否为空。
- 若调用 valid 方法后返回的信息字符串为空串，则通过第 21～23 行代码继续将其他数据封装到对象 u 中，在 24 行调用 insert 方法将数据插入到用户信息表中；若插入失败，将返回的错误信息赋值给变量 msg。
- 第 28 行将前期处理得到的结果信息存储到 request 对象中，以便在后面转发的页面中可以显示此结果信息。
- 29 和 30 行将请求转发到显示结果信息的页面 chkRegResult.jsp。
- 结果显示页面 chkRegResult.jsp 的代码比较简单，如下所示：

主要代码　web\10\10_3\chkRegResult.jsp

```
1: <body>
2:     <%
3:         request.setCharacterEncoding("UTF-8");
4:         String msg = (String) request.getAttribute("msg");
5:         if (msg.equals("")) {
6:             out.println("注册成功！现在将自动转向登录页面...");
7:             response.setHeader("refresh", "3;url=login.jsp");
8:         } else {
9:     %>
10:     <jsp:forward page="reg.jsp">
11:         <jsp:param name="msg" value="<%=msg%>" />
12:     </jsp:forward>
13:     <%}%>
14: </body>
```

该页面的第 4 行代码提取了前一页面写入 request 对象中的属性 msg，其余代码比较简单，这里不再说明。将 reg.jsp 和 chkRegResult.jsp 复制到 web 目录中，访问 reg.jsp，可得到与之前相同的运行结果。

10.3.5　重点提示

- 应根据请求 Servlet 的方式，编写相应的 doXXX 方法。
- 应注意 JSP 的内建对象在 Servlet 中相应的处理方法。
- 掌握在 Servlet 中转发请求的方法。

10.4　使用 Ajax 实现用户名唯一性的即时检测

10.4.1　任务提出

在前面完成的注册功能中，需待用户填写完所有的数据后，才能将数据提交到服务器进行统一的检测，若此时发现用户名已被其他人注册了，则用户必须返回原注册页面重新填写数据。若在新用户输入用户名后，能立即检测出该用户名是否可用，将带给用户更加良好的体验。这一需求可通过 Ajax 技术实现。

10.4.2　Ajax 简介

Ajax 是异步 JavaScript 和 XML（Asynchronous JavaScript and XML）的英文缩写，使用 Ajax 技术可以改善 Web 页面的即时交互性，可以实现页面的局部刷新等。

Ajax 的核心概念在于使用 XMLHttpRequest 对象发送异步请求，所谓异步请求，是相对于传统的 Web 应用的同步交互过程而言的。在传统的 Web 应用中，客户端向服务器发送一个请求，服务器在接收到请求后，去解析客户端所请求的资源，并向用户返回结果。在服务器处理请求的这段期间，用户只能等待。有时，用户仅需要改变页面中的部分数据，却不得不刷新整个页面。为了改善传统 Web 应用中这种不良的用户体验，出现了 Ajax 技术。

10.4.3　Ajax 的基本工作原理

Ajax 主要通过 JavaScript 调用 XmlHttpRequest 对象，向 Web 服务器传送异步请求。服务器处理该请求，在服务器处理用户请求的同时，用户依然可以在原页面进行其他操作。客户端可以通过检测 XmlHttpRequest 对象的状态（readyState 属性），来查看服务器是否已完成请求。当服务器完成处理后，将响应传回客户端，响应的内容可由 XmlHttpRequest 对象的 responseText 属性获取。

10.4.4　任务实现

可按照下面的思路完成用户名重复的即时检测。在注册页面 reg.jsp 中，当用户完成用户名的填写后（即光标离开了用户名输入框），调用 JS 函数向服务器的检测资源（可以是 JSP 页面或其他资源）发起异步请求，待服务器返回检测结果后，根据该结果在页面上显示相应的提示消息。

在 reg.jsp 中，需对用户名输入框添加 blur 事件的处理，并在该页面中添加对相应的 JS 文件的引用，修改或添加的主要代码如下所示：

修改后的代码　web\10\10_4\reg.jsp

```
1：<html>
2：   <head>
3：<!--此处略去了未改变的代码-->
4：      <!--Ajax 检测函数-->
5：      <script src="js/ajax.js"></script>
6：   </head>
7：   <body id="reg" onload="$('uname').focus();">
8：            <form action="<%=request.getContextPath()%>/ChkRegServlet2"
                 method="post" onsubmit="return check()">
9：              <table>
10：                <tbody>
11：                  <tr><td colspan="2" id="title">请填写注册信息</td></tr>
12：                  <tr><td class="ltd"> 用户名* </td>
13：                    <td>
14：                      <input type="text" name="uname" size="40"
                          maxlength="20"    onblur="chkuser()"/>
15：                      <span id="unameinfo"></span>
16：                    </td>
17：          <!--此处省略了其他未改变的代码-->
18：    </body>
19：</html>
```

代码说明如下。

- 这里只列出了有修改的代码。
- 第 5 行添加了相关的 JS 文件的引用。
- 第 8 行修改了 action 属性，提交到另一个 Servlet 去处理。
- 第 14 行为用户名输入框添加了 onblur 的事件处理，当完成了用户名的填写后，将调用 JS 函数 chkuser 检测用户名的可用性。
- 第 15 行添加了一对标签，用于显示检测的结果信息。

JS 文件 ajax.js 中编写了完成异步检测的数个函数，代码如下：

程序清单　web\js\ajax.js

```
1：/*--------当页面载入时，先创建 xmlhttp 异步传输对象---------*/
2：try {
3：    xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
4：} catch (e) {
5：    try {
6：       xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
7：    } catch (e2) {
8：       xmlHttp = false;
9：    }
10：}
11：if (!xmlHttp && typeof XMLHttpRequest != 'undefined') {
12：    xmlHttp = new XMLHttpRequest();
13：}
```

```
14：
15：/*------------------------各函数的定义------------------------*/
16：function $(id)
17：{
18：    return document.getElementById(id);
19：}
20：
21：/*  函数 chkuser，在 reg.jsp 页面中，当用户名输入完毕时被调用
22：    功能：向服务器发起请求，请检查用户名是否重复，并等待查询结果，
23：    当服务器把处理结果传送回来后，调用 updatePageUser 函数继续处理
24：*/
25：function chkuser() {
26：    var username = document.getElementById("uname").value;
27：    if (username == null||username == "")
28：    {
29：        alert("请输入用户名.");
30：        $("uname").focus();
31：        return;
32：    }
33：
34：    /*设置完成后台检测的 url，这里准备向检测用户名
35：      *是否重复的 Servlet 发起请求，
36：      *同时传递请求参数 name(即需要检测的用户名)，
37：      *该参数需经过编码，因为可能含有中文
38：    */
39：    var url = "ChkUniqueName?uname="+encodeURI(username);
40：
41：    /*准备向服务器发起请求，设置请求的 url，
42：      *传递数据的方式为 GET，
43：      *第 3 个参数 true 表示采用异步请求
44：    */
45：    xmlHttp.open("GET", url, true);
46：
47：    //当服务器的状态改变时，由函数 update 进行跟踪处理
48：    xmlHttp.onreadystatechange = update;
49：
50：    //发出请求
51：    xmlHttp.send(null);
52：}
53：
54：/*根据服务器传回的处理结果，决定在客户端的页面中显示什么信息*/
55：function update() {
56：
57：    if (xmlHttp.readyState != 4) {
58：        //请求未完成,在信息提示区显示提示信息
59：        $("unameinfo").innerHTML="正在检测.....";
```

```
60:    }
61:    else  //服务器已完成请求，并传回了响应结果
62:    {
63:      //获取响应的内容，这里是服务器执行 Servlet(ChkUniqueName)后得到的结果
64:      var response = xmlHttp.responseText;
65:      if (response!=""){
66:    //若在 ChkUniqueName.java 中检测后用户名重复，则返回的结果信息不为空
67:          $("unameinfo").innerHTML='<img src="images/check_error.gif"
       width="13" height="13"> 已被使用';
68:          $("uname").focus();  //光标定位在用户名输入框，等待用户重新输入
69:      }else{
70:          $("unameinfo").innerHTML='<img src="images/check_right.gif"
       width="13" height="13">可用';
71:      }
72:    }
73: } //end of function update
```

代码中已有详细的注释，这里仅做一些补充说明。

- 当页面 reg.jsp 被加载时，将首先执行代码第 1～13 行，从而创建一个 XMLHttpRequest 对象，后面将利用该对象向服务器发出异步请求。
- 第 25～52 行编写的 chkuser 函数在 reg.jsp 页面的第 14 行被调用，当用户的输入光标离开了用户名输入框时，将触发 chkuser 函数的执行，该函数利用已经创建的 xmlHttp 对象向服务器的检测资源（即/ChkUniqueName）发出请求，第 48 行指定由函数 update 跟踪此次请求的执行过程。
- 第 55～73 行定义了 update 函数，它通过读取服务器发回的响应状态码（xmlHttp 对象的 readyState 属性），判断是否已经完成本次请求，若已经完成，则读取服务器发回的响应内容，若该内容为空，表示用户名不重复，根据结果在页面的指定位置显示提示信息。

虽然异步请求是由 JS 函数发出的，但真正的“幕后英雄”却是在服务器端执行检测操作的 Servlet，该 Servlet 调用 Users 类的 uniqueName 方法完成了检测工作，代码如下：

程序清单　src\java\chap10\p4

```
1: package chap10.p4;
2:
3: import chap10.p2.Users;
4: import java.io.*;
5: import javax.servlet.ServletException;
6: import javax.servlet.http.*;
7: public class ChkUniqueName extends HttpServlet {
8:    protected void doGet(HttpServletRequest request,
                           HttpServletResponse response)
9:              throws ServletException, IOException {
10:      //保证不被浏览器缓存
11:      response.setHeader("pragma", "no-cache");
12:      response.setHeader("cache-control", "no-cache,must-revalidate");
```

```
13:         response.setHeader("expires", "0");
14:         response.setContentType("text/html;charset=UTF-8");
15:         request.setCharacterEncoding("UTF-8");
16:         Users u = new Users();
17:         u.setUname(request.getParameter("uname"));
18:         String msg = u.uniqueName();
19:         PrintWriter out = response.getWriter();
20:         out.print(msg);
21:         out.flush();
22:         out.close();
23:     }
24: }
```

代码说明如下。

- 该 Servlet 只处理 GET 方式的请求，因此只重写了 doGet 方法。
- 代码第 11～13 行保证了浏览器不缓存此次请求的响应内容，这些代码不能省略，否则有可能引起错误的检测结果。
- 第 16～18 行创建了 Users 类的对象，设置其 uname 属性的值为用户所输入的用户名，然后调用 uniqueName 方法检测此用户名是否重复，返回的结果保存到变量 msg 中。
- 第 19～22 行将检测结果输出到客户端，并关闭输出流。

由于采用了 Ajax 检测用户名的唯一性，因此不必在原来的 ChkRegServlet.java 中再进行同类型的检测。修改该 Servlet，另存为 ChkRegServlet2.java，并在 web.xml 中添加相应的配置段。最后，修改页面 reg.jsp 的 form 标签，使 action 属性指向对 ChkRegServlet2.java 的访问。

将所有的页面复制到 web 目录下，访问 reg.jsp 页面，得到如图 10-12 和图 10-13 所示的运行结果。



图 10-12　用 Ajax 检测用户名重复



图 10-13　用户名可用

在本例中，我们直接通过编写 JS 函数实现了异步的检测。实际上，由于 Ajax 技术的广泛应用，目前有许多 Ajax 框架可供使用，如 DWR 框架等，使用框架可使 Ajax 的应用变得更加简单。

10.5　为用户注册添加头像上传功能

10.5.1　任务提出

在论坛网站中，用户通过虚拟的网络平台进行交流，彼此之间没有更多的了解，为了使用户之间有更感性更直观的认识，往往会由用户为自己设置一个头像。

头像功能一般有两种处理方式：一是由系统设定一系列头像，用户可在其中挑选一个；另一种更个性化的方式是由用户自己提供图片文件作为自己的头像，此时则需要将用户选定的图片文件从用户机器上传到服务器机器上。

此外，我们也经常在网络中上传或下载资料，文件上传是网站常用的基本功能。

我们可以通过 Java 中提供的文件操作功能编写自己的上传程序，但是目前已有很多成熟的上传组件可供我们使用，如 Apache 的 Commons FileUpload、jspSmartUpload 等。

jspSmartUpload 是由 www.jspsmart.com 网站提供的一个可免费使用的全功能的文件上传下载组件，适于嵌入执行上传下载操作的 JSP 文件中。本节将介绍使用该组件上传文件的方法。

10.5.2　jspSmartUpload 的相关类

1．File 类

该类封装了一个上传文件的所有信息，如上传文件的文件名、文件大小、扩展名、文件数据等。File 类主要提供的方法如表 10-1 所示。

表 10-1　File 类的主要方法

序　号	方 法 名	作　用	原　型
1	saveAs	将文件换名另存	public void saveAs(String destFilePathName) 或 public void saveAs(String destFilePathName, int optionSaveAs)
2	isMissing	判断用户是否选择了文件，选择了文件时，返回 false，未选文件时，返回 true	public boolean isMissing()
3	getFileName	取文件名(不含目录信息)	public String getFileName()
4	getFilePathName	取文件全名(含目录信息)	public String getFilePathName()
5	getFileExt	取文件扩展名（后缀）	public String getFileExt()
6	getSize	取文件长度（以字节计）	public int getSize()

saveAs 方法的补充说明如下。

该方法的作用是将上传文件换名后另存，其第 2 个参数用以说明第 1 个参数（即上传文件）的存储路径的类型，可取 3 个值，分别为 SAVEAS_PHYSICAL、SAVEAS_VIRTUAL 和 SAVEAS_AUTO。

- SAVEAS_PHYSICAL：以操作系统的根目录为文件根目录来另存文件。
- SAVEAS_VIRTUAL：以 Web 应用程序的根目录为文件根目录来另存文件。
- SAVEAS_AUTO：让组件以下面的方式决定路径的类型，即当 Web 应用程序的根目录下存在由第 1 个参数指定的目录时，以 SAVEAS_VIRTUAL 方式处理，否则按照 SAVEAS_PHYSICAL 方式处理。

例如，执行“saveAs("/upload/sample.zip",SAVEAS_PHYSICAL)”时，若 Web 服务器安装在 C 盘，则另存的文件路径实际是 C:\upload\sample.zip。

而执行“saveAs("/upload/sample.zip",SAVEAS_VIRTUAL)”执行时，若 Web 应用程序的根目录是 D:\Tomcat 6.0\webapps\jspbook，则上传文件的实际存储路径为 D:\Tomcat 6.0\webapps\jspbook\upload\sample.zip。

执行“saveAs("/upload/sample.zip",SAVEAS_AUTO)”时，若 Web 应用程序根目录下存在 upload 目录，则其效果同“saveAs("/upload/sample.zip",SAVEAS_VIRTUAL)”，否则同“saveAs("/upload/sample.zip",SAVEAS_PHYSICAL)”。

2．Files 类

该类表示所有上传文件的集合，通过它可以得到上传文件的数目、大小等信息，其方法如表 10-2 所示。

表 10-2　Files 类的主要方法

序　号	方 法 名	作　用	原　型
1	getCount	取得上传文件的数目	public int getCount()
2	getFile	取得指定序号的 com.jspsmart.upload.File 文件对象	public File getFile(int index) 其中 index 为指定的序号，其值在 0 到 getCount()-1 之间
3	getSize	取得上传文件的总长度，可用于限制一次性上传的数据量大小	public long getSize()

3．Request 类

这个类的功能等同于 JSP 内置的对象 request。之所以提供这个类，是因为对于文件上传表单，通过 request 对象无法获得表单项的值，必须通过 jspSmartUpload 组件提供的 Request 对象来获取。该类提供的方法如表 10-3 所示。

表 10-3　Request 类的主要方法

序　号	方 法 名	作　用	原　型
1	getParameter	获取指定参数的值。当参数不存在时，返回值为 null	public String getParameter(String name) 其中，name 为参数的名字
2	getParameterValues	当一个参数可以有多个值时，用此方法来取其值。它返回的是一个字符串数组。当参数不存在时，返回值为 null	public String[] getParameterValues(String name)
3	getParameterNames	取得Request对象中所有参数的名字，用于遍历所有参数。它返回的是一个枚举型的对象	public Enumeration getParameterNames()

4．SmartUpload 类

这个类完成的是上传下载的工作，其主要方法如表 10-4 所示。

表 10-4　SmartUpload 类的主要方法

序　号	方 法 名	作　用	原　型
1	initialize	执行上传下载的初始化工作，必须第一个执行	最常使用的是： public final void initialize(javax.servlet.jsp.PageContext pageContext) 其中，pageContext 为 JSP 页面内置对象
2	upload	上传文件数据	public void upload()
3	save	将全部上传文件保存到指定目录下，并返回保存的文件个数	public int save(String destPathName) 和 public int save(String destPathName,int option) 注：save(destPathName) 作用等同于 save(destPathName,SAVE_AUTO)
4	getSize	取上传文件数据的总长度	public int getSize()
5	getFiles	取全部上传文件，以 Files 对象形式返回，可以利用 Files 类的操作方法来获得上传文件的数目等信息	public Files getFiles()
6	getRequest	取得 Request 对象，以便由此对象获得上传表单参数的值	public Request getRequest()

续表

序　号	方 法 名	作　用	原　型
7	setAllowedFilesList	设定允许上传带有指定扩展名的文件，当上传过程中有文件名不允许时，组件将抛出异常	public void setAllowedFilesList(String allowedFilesList) 其中，allowedFilesList 为允许上传的文件扩展名列表，各个扩展名之间以逗号分隔。如果想允许上传那些没有扩展名的文件，可以用两个逗号表示
8	setDeniedFilesList	用于限制上传那些带有指定扩展名的文件。若有文件扩展名被限制，则上传时组件将抛出异常	public void setDeniedFilesList(String deniedFilesList)
9	setMaxFileSize	设定每个文件允许上传的最大长度	public void setMaxFileSize(long maxFileSize) 其中，maxFileSize 为为每个文件允许上传的最大长度，当文件超出此长度时，将不被上传
10	setTotalMaxFileSize	设定允许上传的文件的总长度，用于限制一次性上传的数据量大小	public void setTotalMaxFileSize(long totalMaxFileSize)
11	setContentDisposition	下载文件时使用，将数据追加到 MIME 文件头的 CONTENT-DISPOSITION 域	public void setContentDisposition(String contentDisposition)
12	downloadFile	下载指定的文件	最常用的： public void downloadFile(String sourceFilePathName) 其中，sourceFilePathName 为要下载的文件名（带目录的文件全名）

10.5.3　使用 jspSmartUpload 上传文件

首先，应将 jspSmartUpload 组件的 jar 文件复制到项目的 WEB-INF\lib 目录或 Tomcat 的 lib 目录下。

若是在 NetBeans 中编写这个程序，则还需在项目窗口中右击“库”，选择 jspSmartUpload 的 jar 文件。添加后的结果如图 10-14 所示。



图 10-14　在 NetBeans 中添加 jar

【例 10.4】　文件上传。

在页面 fileSelect.jsp 中选择需要上传的文件，提交至 upload.jsp，由该页面提取上传的文件内容，并将文件保存在服务器端，具体位置为项目根目录下的 UserFiles\Images 目录下，若该目录不存在，将自动创建；而上传后保存的文件名则由工具类 FileUtil 随机生成，以保证来自不同客户端的文件名不重复。

主要代码　web\10\10_5\jspSmart\fileSelect.jsp

```
1:      <body>
2:          <% String err = request.getParameter("err");
```

```
3:          if (err == null) err = "";
4:        %>
5:        <div><%= err%></div>
6:        <form   name="frm1" id="frm1" enctype="multipart/form-data"
          action="upload.jsp" method="post">
7:          上传者：<input type="text" name="username" /><br/>
8:          文  件：
9:          <input type="file" name="filename" size="30"/><br/>
10:           <input type="submit" value="确定" />
11:           <input type="reset" value="重置" />
12:         </form>
13:       </body>
```

这个页面主要提供上传文件的选择。

- 代码第 6～12 行提供了文件选择表单，与以往收集普通数据的表单有所不同，第 6 行中 form 标签的 enctype 属性需设为“multipart/form-data”，表示需要上传文件，并且数据传送的方式应设为 post。
- 若提交到 upload.jsp 页面后，发现文件大小超过规定，或文件类型不是指定的类型时，会重新回到这个选择页面，并且利用请求参数 err 将异常信息带回。因此，本页面的第 2～5 行代码用于提取和显示这个错误提示信息。

程序清单　文件 web\10\10_5\jspSmart\upload.jsp

```
1: <body>
2: <%
3: String myFileName = "";
4: String username = "";
5: try {
6:   //创建上传组件实例
7:   SmartUpload mySmartUpload = new SmartUpload();
8:   //初始化上传组件
9:   mySmartUpload.initialize(pageContext);
10:  //设置被上传文件的最大长度限制
11:  mySmartUpload.setMaxFileSize(500 * 1024);
12:  //设置允许上传的文件的类型
13:  mySmartUpload.setAllowedFilesList("jpg,gif,bmp,png,doc,xls");
14:  //上传文件
15:  mySmartUpload.upload();
16:  //提取表单中的其他数据项，这里是上传者的用户名
17:  username = mySmartUpload.getRequest().getParameter("username");
18:  //获取上传的第一个文件对象
19:  com.jspsmart.upload.File myFile = mySmartUpload.getFiles().getFile(0);
20:  //判断文件对象是否存在
21:  if (myFile != null && !myFile.isMissing()) {
22:    //取得上传的文件的文件名
23:    myFileName = myFile.getFileName();
24:    //上传文件的保存目录
```

```
25:        String sdir = application.getRealPath("/") + "UserFiles\\Images\\";
26:        //检测目录是否存在
27:        java.io.File dir = new java.io.File(sdir);
28:        if (!dir.exists()) {
29:            dir.mkdirs();     //若目录不存在，则自动创建
30:        }
31:        //文件保存的绝对路径
32:        String newFileName = sdir +
     FileUtil.createFileName(myFile.getFileExt());
33:        //以绝对路径形式保存文件，以文件系统的根目录为参照
34:        myFile.saveAs(newFileName, SmartUpload.SAVE_PHYSICAL);
35:        //输出提示信息
36:        out.println("恭喜" + username + ",文件" + myFileName +
          "已经上传成功，<br/>被保存为" + newFileName);
37:     } else {
38:        response.sendRedirect("fileSelect.jsp?err=" +
             URLEncoder.encode("请选择文件.", "utf-8"));
39:     }
40: } catch (SecurityException e) {
41:     response.sendRedirect("fileSelect.jsp?err=" +
  URLEncoder.encode("只能上传 500K 内的指定类型的文件.", "utf-8"));
42: } catch (Exception e) {
43:     e.printStackTrace(System.err);
44: }
45: %>
46: </body>
```

代码中已有较详细的注释，这里仅做些补充。

- 第 17 行的 mySmartUpload.getRequest()得到一个 SmartUpload 的 Request 对象，该对象与内置对象 request 相似，其 getParameter 方法可以提取非文件类型的表单数据项。注意，这里使用内置对象 request 并不能得到表单项的值。
- 第 19 行代码中，mySmartUpload.getFiles()得到一个 Files 对象，表示所有上传文件的集合，因为 SmartUpload 组件允许一次上传多个文件。而 Files 的 getfile(i)方法表示根据索引号 i 得到集合中的某个 File 对象。这个例子中只上传了一个文件，因此可以用 getFile(0)得到该文件对象。
- 第 25 行的 application.getRealPath("/")表示获取与应用根目录对应的绝对路径。例如，项目 jspbook 已发布在 D:\Tomcat 6.0\webapps\下，则表达式 application.getRealPath("/")得到的路径为 D:\Tomcat 6.0\webapps\jspbook\。
- 第 32 行的 myFile.getFileExt()获取了上传文件的文件名后缀。而代码 FileUtil.createFileName()是调用自定义的 FileUtil 类的 createFileName()方法，根据给定的后缀生成一个随机的文件名，该文件名根据当前时间和随机数生成。
- 第 34 行代码调用了组件的 saveAs 方法，将 File 对象以指定的文件名保存在指定的目录下。
- 案例中还使用了 FileUtil 类，以便随机生成一个不容易重复的文件名。在完成文件上传

的功能时，若需要将来自不同客户端的大量上传文件存放在同一个文件夹中，则一般需要将上传的文件重新命名，避免文件因同名而被覆盖。这里的 FileUtil 类提供了一种根据当前时间随机生成文件名的解决方案。

程序清单　文件 src\chap10\FileUtil.java

```
1：package chap10;
2：
3：import java.text.SimpleDateFormat;
4：public class FileUtil {
5：        public static synchronized String createFileName() {
6：                return createFileName("txt");   //默认后缀名为 txt
7：        }
8：// Function: 生成随机文件名
9：        public static synchronized String createFileName(String ext) {
10：                if (ext == null) {
11：                        ext = "txt";
12：                }
13：                java.util.Date now = new java.util.Date();
14：                SimpleDateFormat sdf = new SimpleDateFormat("yyMMddHHmmssSSS");
15：                String fileName = sdf.format(now);
16：                int rnd = (int) Math.round(Math.random() * 10000);
17：                StringBuffer s = new StringBuffer("0000").append(rnd);
18：                s.delete(0, s.length() - 4);
19：                fileName = fileName + s.toString() + "." + ext;
20：                return fileName;
21：        }
22：}
```

运行效果如图 10-15 和图 10-16 所示。

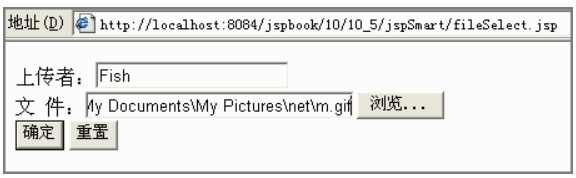


图 10-15　选择上传文件

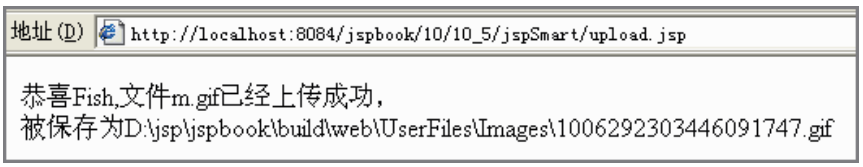


图 10-16　文件上传成功

10.5.4　任务实现

要提供头像选择或上传功能，首先需要修改注册输入页面 reg.jsp，将头像设置部分的代码修改为：

修改的代码　web\10\10_5\reg.jsp

```
1：<tr><td class="ltd">选择头像</td>
2：    <td style="vertical-align:middle">
3：    <input type="hidden" name="ulogo" id="ulogo"/>
4：    <img src="images/head/space_noface.gif" id="hlogo" name="hlogo" /><br>
5：    <input type="radio" name="s1" onclick="selectImg()"/>选择
6：    <input type="radio" name="s1" onclick="uploadImg()"/>上传
7：    </td>
8：</tr>
```

同时需要修改 JS 文件 reg.js，为其添加 selectImg 函数和 uploadImg 函数，分别用于打开头像选择窗口和上传文件选择窗口，相关代码如下：

添加的函数代码　web\js\reg.js

```
1：function selectImg(){
2：    $('ulogo').value=window.showModalDialog('selectHead.jsp',
                    window,'dialogHeight:320px;dialogWidth:450px;
                     dialogTop:200px;dialogLeft:600px;center:Yes;
                     help:No;resizable:Yes;status:No;');
3：    $('hlogo').src=$('ulogo').value;
4：}
5：function uploadImg(){
6：    $('ulogo').value=window.showModalDialog('uploadImg.jsp',
                     window,'dialogHeight:280px;dialogWidth:320px;
                      dialogTop:200px;dialogLeft:600px;center:Yes;
            help:No;resizable:Yes;status:No;');
7：    $('hlogo').src=$('ulogo').value;
8：}
```

代码说明如下。

- 函数 selectImg 用于打开头像选择窗口，这里利用 JavaScript 中 window 对象的 showModalDialog 函数打开一个模态对话窗口，在此对话窗口中显示页面 selectHead.jsp 的内容，隐藏字段 ulogo 的值设为此对话窗口的返回值，即用户在页面 selectHead.jsp 中所选择的头像图片的文件名。
- 函数 uploadImg 的处理过程与 selectImg 相似，只是打开的不是头像选择对话窗口，而是文件上传对话窗口，在此窗口中显示的是上传页面 uploadImg.jsp 的内容，上传完成后，隐藏域 ulogo 的值被设为上传的图片文件的名称。

因本章节主要演示文件上传组件的使用，故头像选择页面 selectHead.jsp 的代码不在此列出，需要的读者可参照本书所附的源码。

选择上传文件的页面 uploadImg.jsp 的代码如下：

主要代码　web\10\10_5\uploadImg.jsp

```
1：<body style="text-align:center;margin:auto">
2：<div >
3：  <%
4：if (request.getParameter("uploadFilename") == null) {
5：       if (request.getParameter("err") != null) {
6：      String msg=URLDecoder.decode(request.getParameter("err"), "utf-8");
7：                    out.print("<p style='color:red'>"+msg+"</p>");
8：        }
9：   %>
10：    <p> </p>
11：    <p>请选择您要上传的头像图片：</p>
12：    <form   name="frm1" id="frm1" enctype="multipart/form-data"
  action="uploadImgServlet" method="post">
13：      <input type="file"   name="himg">
14：      <input type="button" onclick="if (document.getElementById('himg')
            .value!='') document.frm1.submit();" value="上传"/>
15：      <input type="button" value="取消" onclick="window.returnValue=
     'images/head/space_noface.gif';window.close()"/>
16：           </form>
17：           <%
18：      } else {
19：     String filename = request.getParameter("uploadFilename");
20：      int last = filename.lastIndexOf('.');
21：     filename =   URLDecoder.decode(filename.substring(0, last),
       "utf-8") + filename.substring(last);
22：      out.print("<script> window.returnValue=
 'UserFiles/Image/"+filename+"';window.close();</script>");
23：      }
24：           %>
25：      </div>
26：  </body>
```

代码说明如下。

- 该页面通过第 12～16 行代码提供文件选择表单，并提交到 uploadImgServlet 完成上传处理。
- 当 uploadServlet 完成上传处理后，将把处理结果（上传成功时的文件名或上传失败时的出错信息）发送回本页面；若本页面能接收到请求参数 uploadFilename，则表示上传成功，此时通过 19～23 行代码将上传文件路径作为本页面所在窗口的返回值，该返回值将在 reg.js 中的 JS 函数 uploadImg 中进行处理，成为输入页面 reg.jsp 中隐藏字段 ulogo 的值。
- 若未能接收到请求参数 uploadFilename，表示未进行上传或上传失败，则通过第 5～8 行代码决定是否需要显示上传失败的提示信息。

完成上传功能的 Servlet 的代码与前面上传案例中 upload.jsp 页面基本相似，代码如下：

程序清单　src\java\chap10\p5\UploadImgServlet.java

```
1：package chap10.p5;
2：
3：import com.jspsmart.upload.*;
4：import java.io.IOException;
5：import java.net.URLEncoder;
6：import java.sql.SQLException;
7：import javax.servlet.ServletException;
8：import javax.servlet.http.*;
9：
10：public class UploadImgServlet extends HttpServlet {
11：    @Override
12：    protected void doPost(HttpServletRequest request,
                                HttpServletResponse response)
13：    throws ServletException, IOException {
14：    response.setContentType("text/html,charset=UTF-8");
15:            request.setCharacterEncoding("utf-8");
16：try {
17：      SmartUpload su= new SmartUpload();
18：      su.initialize(this,request,response);
19：      su.setMaxFileSize(200 * 1024);
20：      su.setAllowedFilesList("jpg,jpeg,gif,bmp,png,JPG,
    JPEG,GIF,BMP,PNG");
21：      su.upload();
22：      com.jspsmart.upload.File myFile = su.getFiles().getFile(0);
23：      if (myFile != null && !myFile.isMissing()) {
24：          String newFilename =   FileUtil.
                    createFileName(myFile.getFileExt());
25：            myFile.saveAs("UserFiles/Image/" + newFilename,
              SmartUpload.SAVE_VIRTUAL);
26：            response.sendRedirect("uploadImg.jsp?uploadFilename="
            + newFilename);
27：      }else{
28：            response.sendRedirect("uploadImg.jsp?err=" +
            URLEncoder.encode("请选择文件.", "utf-8"));
29：      }
30：      } catch (SmartUploadException ex) {
31：                response.sendRedirect("uploadImg.jsp?err=" +
                    URLEncoder.encode("上传失败!", "utf-8"));
32：    } catch (SecurityException e) {
33：                response.sendRedirect("uploadImg.jsp?err=" +
        URLEncoder.encode("只能上传 200K 以内的图像文件.", "utf-8"));
34：      } catch (SQLException e) {
35：                response.sendRedirect("uploadImg.jsp?err=" +
            URLEncoder.encode("上传失败!", "utf-8"));
36：            }
```

```
37:        }
38: }
```

该 Servlet 类的处理过程与页面 upload.jsp 基本相似，下面仅做简单的补充说明。

- 第 18 行使用对组件进行初始化。
- 第 20 行指定所允许上传的文件的类型。
- 第 25 行通过指定虚拟路径的形式将上传文件保存到当前应用的 UserFiles/Image 目录下，因此应保证相关目录已经创建，否则会导致上传失败。
- 第 26 行表示上传成功时，重定向到 uploadImg.jsp，即原来的文件选择窗口，并将文件名作为参数传给该页面。
- 第 30～36 行表示发生各种异常时，将相应的异常信息作为参数，重定向到原来的文件选择页面 uploadImg.jsp。

至此，注册页面中的头像上传功能已经完成，将所有页面复制到 web 目录下，访问 reg.jsp，单击其中的“选择”或“上传”单选按钮时，弹出的对话窗口如图 10-17 和图 10-18 所示。



图 10-17　头像选择对话窗口

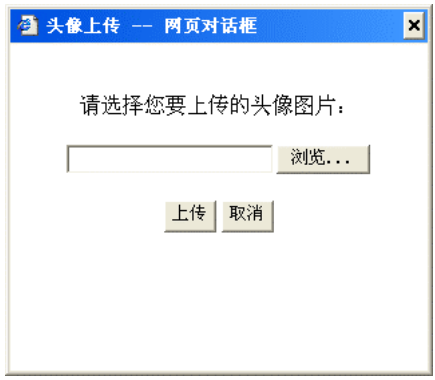


图 10-18　上传文件选择窗口

当选择其中的某个头像并单击“确定”按钮后，注册页面显示结果如图 10-19 所示。若在上传文件选择窗口中选择并上传一个图片文件作为头像，则注册页面显示如图 10-20 所示的结果。



图 10-19　选择了一个系统头像



图 10-20　上传自定义的头像

10.5.5　重点提示

- 提供文件上传的表单，应在 form 标签中添加 enctype 属性，并将该属性值设为“multipart/form-data”，同时，表单的数据提交方式应设为“post”。
- 使用 jspSmartUpload 组件上传文件时，普通的表单数据（非文件）不能直接通过内置对象 request 获取。
- JavaScript 中可通过 window 对象的 openModalDialog 函数打开模态对话窗口，该窗口可有返回值，由 returnValue 属性设置。

10.6　为注册输入页面添加验证码

为了防止被人使用恶意软件完成数据的填写，往往会在页面中使用验证码来验证用户填写数据的即时性。

生成验证码的方法很多，这里只介绍生成 4 位整数的随机验证码的方法。首先，可以利用随机数类 Random 来产生所需的随机数，再利用图形类生成相关的图片。

程序清单　web\10\10_6\image.jsp

```
1：<%@ page contentType="image/jpeg" pageEncoding="UTF-8"
 import="java.awt.*,java.awt.image.*,java.util.*,javax.imageio.*" %>
2：<%!
3：Color getRandColor(int fc,int bc){//给定范围获得随机颜色
4：        Random random = new Random();
5：        if(fc>255) fc=255;
6：        if(bc>255) bc=255;
7：        int r=fc+random.nextInt(bc-fc);
8：        int g=fc+random.nextInt(bc-fc);
9：        int b=fc+random.nextInt(bc-fc);
10：         return new Color(r,g,b);
11：         }
12：%>
13：<%
14：//设置页面不缓存
15：response.setHeader("Pragma","No-cache");
16：response.setHeader("Cache-Control","no-cache");
17：response.setDateHeader("Expires", 0);
18：session.removeAttribute("srand");
19：// 在内存中创建图象
20：int width=60, height=20;
21：BufferedImage image = new BufferedImage(width, height,
                                BufferedImage.TYPE_INT_RGB);
22：
23：// 获取图形上下文
24：Graphics g = image.getGraphics();
25：
```

```
26：//生成随机类
27：Random random = new Random();
28：
29：// 设定背景色
30：g.setColor(getRandColor(200,250));
31：g.fillRect(0, 0, width, height);
32：
33：//设定字体
34：g.setFont(new Font("Times New Roman",Font.PLAIN,18));
35：
36：// 随机产生 155 条干扰线，使图像中的认证码不易被其他程序探测到
37：g.setColor(getRandColor(160,200));
38：for (int i=0;i<155;i++)
39：{
40：  int x = random.nextInt(width);
41：  int y = random.nextInt(height);
42：        int xl = random.nextInt(12);
43：        int yl = random.nextInt(12);
44：  g.drawLine(x,y,x+xl,y+yl);
45：}
46：
47：// 取随机产生的认证码(4 位数字)
48：String sRand="";
49：for (int i=0;i<4;i++){
50：      String rand=String.valueOf(random.nextInt(10));
51：      sRand+=rand;
52：      // 将认证码显示到图像中
53：      g.setColor(new Color(20+random.nextInt(110),
    20+random.nextInt(110),20+random.nextInt(110)));
54：      g.drawString(rand,13*i+6,16);
55：}
56：// 将认证码存入 SESSION
57：session.setAttribute("rand",sRand);
58：// 图像生效
59：g.dispose();
60：// 输出图像到页面
61：ImageIO.write(image, "JPEG", response.getOutputStream());
62：%>
```

该页面将产生一个随机数验证码，并将该验证码保存到 session 中（第 61 行代码），方便后面的验证处理。

注意：第 1 行设置了这个 JSP 页面的内容格式为“image/jpeg”，表示将产生一个 JPEG 格式的图片文件。

页面 test.jsp 演示了如何使用前面生成的验证码图片，该程序中的关键代码是：

```
<img border=0 src="image.jsp" />
```

test.jsp 页面的运行效果如图 10-21 所示，这里显示的验证码图片即前面 image.jsp 生成的结果。



图 10-21　验证码的使用

根据 image.jsp 页面中展示的验证码的生成原理，也可以编写生成随机汉字或英文字母的验证图片。

按照上面介绍的方法，很容易在网站的注册页面上添加对验证码的校验。具体代码可参考所附源码的 web\10\10_6\reg.jsp 和 src\java\chap10\p6 中的相关文件。效果如图 10-22 所示。



图 10-22　注册页面的验证码



思考与训练

一、基础训练

1. 请创建一个实体类 Book，包名为 com.aaa.model，含私有属性 bno（String 类型）和 bprice（float 类型），为所有的属性提供 setter 和 getter 方法。

2．请创建一个操作类 BookDAO，包名为 com.aaa.dao，成员如下。

（1）私有变量 oneBook，为 Book 类的对象。

（2）oneBook 的 setter 和 getter 方法。

（3）公有方法 add，返回值类型为 int，作用是将 oneBook 保存到数据库中，具体处理过程可省略。

3．写出<jsp:useBean>动作，创建前面第 1 题中的 Book 类的实例，id 名称为 b1，有效范围为 page，并写出<jsp:setProperty>动作，设置该实例的 bno 属性值为“001”。

4．写出<jsp:useBean>动作，class 为 Users，id 名称为 u1，有效范围为 session，使用<jsp:getProperty>动作显示该实例的 username 属性的值。

5．编写 Servlet，包名为 myservlet，类名为 PrintServlet，在其 doPost 方法中输出自己的姓名。

6．编写上传文件的表单，文件选择框名称为 myphoto。

二、思考题

1．<jsp:useBean>动作的 scope 属性有哪几种取值？分别代表什么含义？

2．使用<jsp:useBean>动作是否一定会创建一个新的实例？

3．在 Servlet 中如何取得与客户端相关的会话对象？如何向客户端输出信息？

4．在 Servlet 中如何实现请求的转发或资源的重定向？

5．一个 Servlet 的生命周期分为哪几个阶段？分别完成哪些工作？

6．在提供文件上传的表单中，form 标签应设置哪个属性？

三、综合实训

1．在发帖功能中添加上传附件功能。

2．改用 Servlet 技术完成论坛的用户登录功能。

3．在登录功能中增加验证码的输入与验证。

第 11 章　服务器的配置与网站的发布

当我们完成网站的开发工作后，需要将项目发布到服务器上。服务器端需要安装 JDK 和一个 Web 服务器软件。本章我们将学习如何配置一个简单的 JSP 网站服务器，并将开发好的项目发布到服务器上。

目前可以作为 Java Web 服务器的软件有很多，如 Tomcat、Resin 等，其中使用最广泛的是 Apache 组织开发的 Tomcat。这是一款免费的开源软件，由于它的使用非常方便，占用的资源也很少，成为了许多开发者的首选。

11.1　Tomcat 的安装与配置

在安装 Tomcat 之前，应该先安装好 JDK，通常按默认的设置安装即可。

11.1.1　Tomcat 的下载

目前 Tomcat 的最新版本是 6.x，大家可以在 Apache 的官方网站上下载安装软件，网址是 http://tomcat.apache.org。

11.1.2　Tomcat 的安装

双击 Tomcat 安装软件，在弹出的欢迎窗口中单击“Next”按钮，然后在下一个对话窗口中单击“I Agree”按钮，表示同意版权协议。接下来将弹出组件选择窗口，一般保持默认状态即可，单击“Next”按钮，进入如图 11-1 所示的对话窗口。

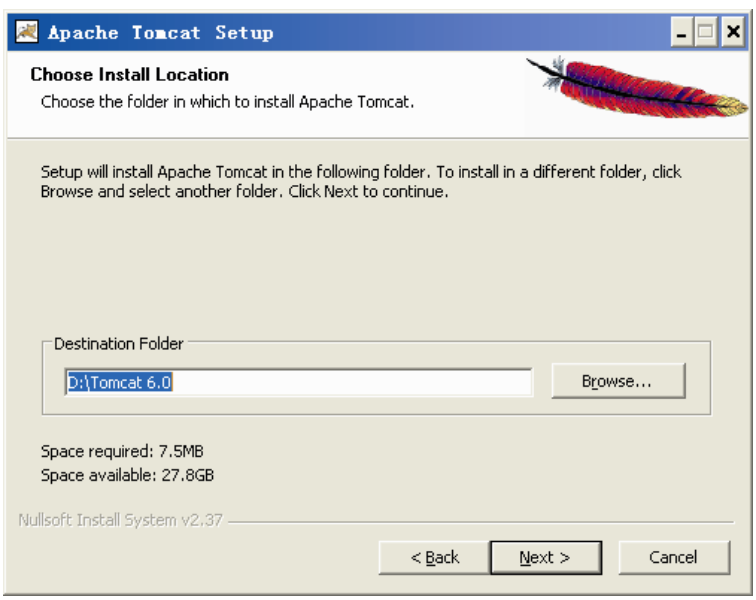


图 11-1　设置安装路径

这里设定 Tomcat 的安装路径，可以通过单击右侧的“Browse”按钮选择不同的安装路径，单击“Next”按钮继续。

在图 11-2 所示的窗口中，第一个输入框中应输入 Tomcat 运行时的端口号，默认的是 8080，若有冲突，可改为其他值；其下的两个输入框为进入 Tomcat 管理界面的用户名和密码，可以自由设定。单击“Next”按钮继续，将弹出如图 11-3 所示的对话窗口。

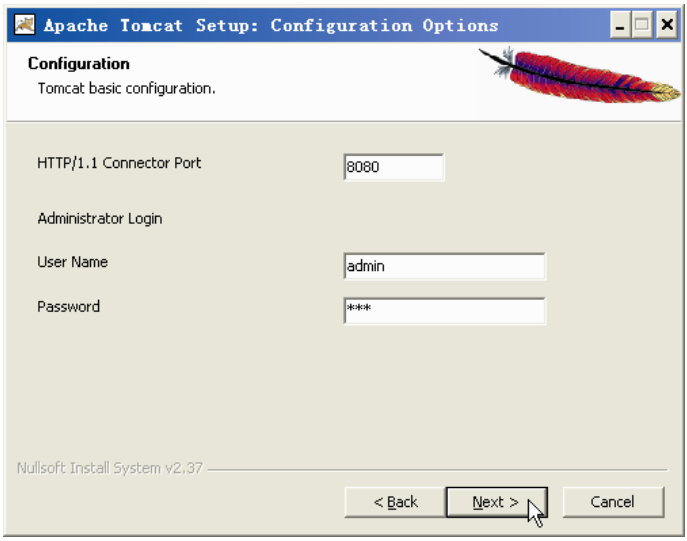


图 11-2　Tomcat 的设置——运行端口及管理密码

若之前已经正确安装了 JDK，则安装软件会自动检测到 JRE 的安装路径，一般是正确的，不需要修改。

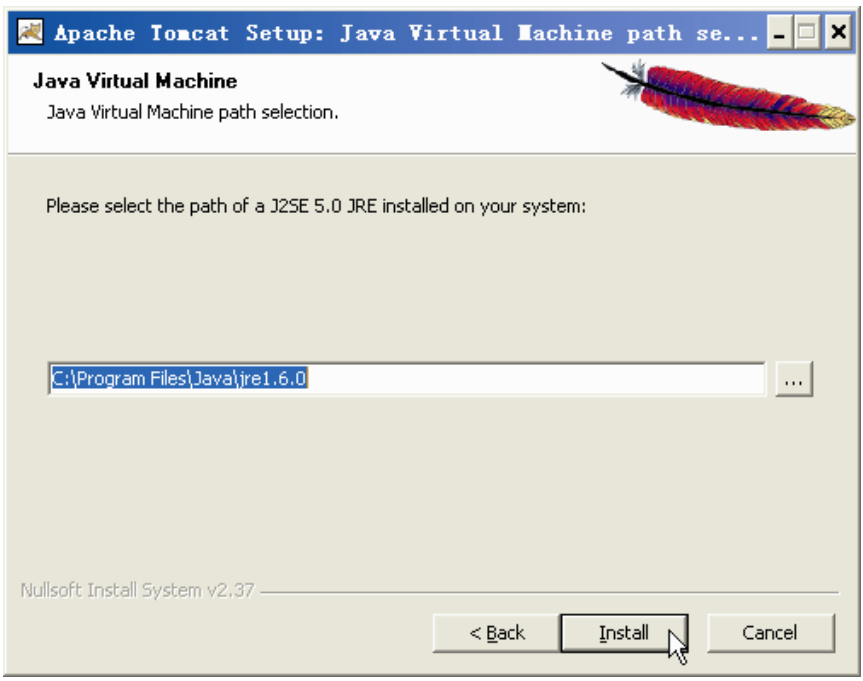


图 11-3　JRE 的安装路径

单击“Install”按钮开始安装，直到出现如图 11-4 所示的窗口，则安装完成。

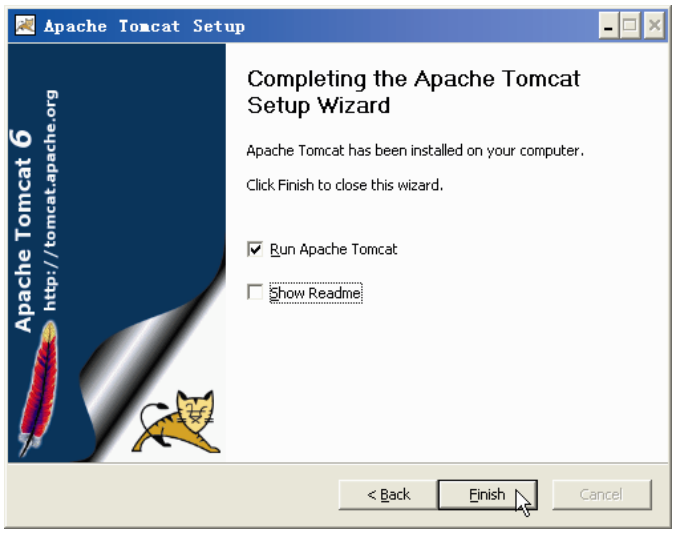


图 11-4　安装完成

窗口中的第一个复选框表示马上启动 Tomcat 服务器，单击“Finish”按钮完成安装。可以看到 Tomcat 已经自动启动，并在 Windows 任务栏的右下角出现了图标，表示服务器已启动。

11.1.3　Tomcat 的启动与停止

1．停止服务器

当 Tomcat 安装完毕后，已经自动启动了服务器。右击 Windows 任务栏右下角的 Tomcat 运行时图标，将出现如图 11-5 所示的菜单，可以在其中选择“Stop service”，停止服务器的运行，相应地，任务栏上的图标变成了。需要注意的是，图 11-5 中所示的“Exit”选项只是退出 Tomcat 的监视工具，并非停止服务器。

2．启动服务器

在 Windows 的“开始”菜单中选择“程序”，可以找到如图 11-6 所示的 Apache Tomcat 6.0 程序组，单击其中的“Monitor Tomcat”，将启动 Tomcat 监视工具，任务栏中右下角出现图标。

右击该图标，在弹出的右键菜单中选择“Start service”，则将启动服务器的运行。



图 11-5　右键菜单

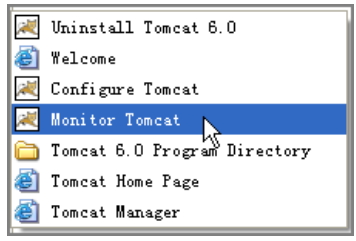


图 11-6　Tomcat 程序组

11.1.4　测试 Tomcat

启动了 Tomcat 服务器后，打开浏览器，输入 http://localhost:8080，若出现如图 11-7 所示

的显示页面，则表示 Tomcat 安装成功。

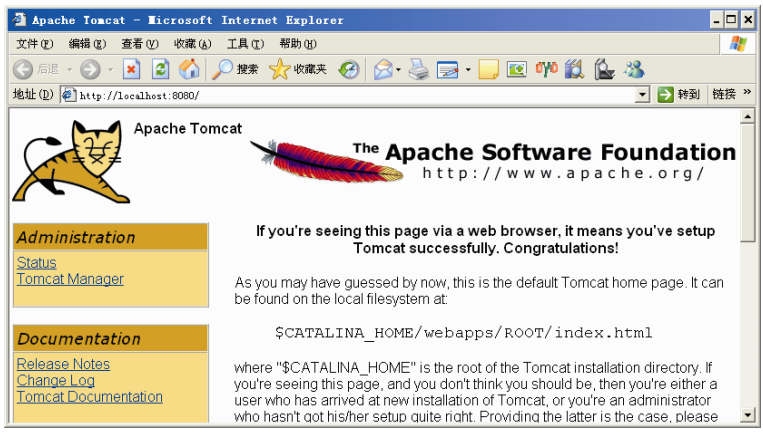


图 11-7　Tomcat 安装成功的测试页面

下面进一步测试服务器的运行环境。

编辑 JSP 文件 t1.jsp，在<body>和</body>间加入代码<% out.print("hello");%>，将该文件保存在 Tomcat 安装目录下面的 webapps\ROOT 下。

在浏览器中输入 http://localhost:8080/t1.jsp，若在浏览器中出现文字 hello，则说明服务器运行正常。

11.2　Tomcat 的目录结构

打开 Tomcat 的安装目录，可以看到其中有很多目录，这里简单介绍一下各目录的作用。

- bin 目录：存放与 Tomcat 运行有关的文件。
- webapps 目录：存放所有布置在该服务器上的 Web 应用。该目录下可以创建多个子目录，不同的应用存放在不同的子目录中。也可以把整个应用打包成.war 压缩文件，Tomcat 启动时会把它自动解压成对应的子目录。
- work 目录：存放编译 JSP 文件后生成的.java 文件和.class 文件。
- logs 目录：存放运行时产生的日志文件。
- conf 目录：存放与服务器相关的配置文件，如 server.xml、context.xml 等。
- lib 目录：存放类库文件，所有布署在此服务器中的应用都可使用这些类库。

11.3　配置网站的异常处理

若程序中存在一些异常而不进行任何处理，则运行时会将异常信息发送到客户端的浏览器中，这时除了带给客户不良的使用体验外，更糟的是会泄露程序代码、数据库结构等信息，从而带来更严重的安全问题。因此，必须重视网站的异常处理，在发布网站前应为其配置合理的异常处理方案。

利用 page 指令，可以独立设置某个页面的异常处理页面，而通过项目的 web.xml 配置文件，可以根据 Java 的异常类型或 HTTP 错误号来决定相应的处理页面，对整个网站的异常情况进行全局的处理。

首先，可以编写一个统一的错误处理页面，如 err.jsp，当网站发生异常时，转向此处理页面。

主要代码　web\11\err.jsp

```
1：<%@ page isErrorPage="true" %>
2：<html>
3：<!-- 此处略去其他代码-->
4：<body>
5：    <%
6：        out.print("页面异常,请稍后再试. ");
7：        exception.printStackTrace(System.err);
8：     %>
9：  </body>
10：</html>
```

其后，在 NetBeans 中打开项目的 Web 页中 WEB-INF 目录下的 web.xml 文件，其中设置了本网站的配置信息。

在如图 11-8 所示的界面中，单击“错误页面”下面的“添加”按钮。



图 11-8　web.xml 的配置界面

在弹出的对话框中，设置需要处理的异常的类型，以及处理这些异常的页面，如图 11-9 所示。



图 11-9　根据异常类型配置错误处理页面

注意：在指定异常类型时，必须指出该异常类包括包名在内的完整类名。

当然，也可以通过错误码来设定对应的处理页面，如图 11-10 所示表示发生了 404 号错误时，由错误处理页面 building.jsp 进行处理。



图 11-10　根据错误代码配置错误处理页面

常见的 HTTP 错误代码的含义如下：

- 404
 The server has not found anything matching the requested URI
 没有找到指定的资源
- 410
 Resource is no longer available.
 资源不可用
- 500
 Internal server error.
 服务器内部错误

在如图 11-8 所示的设置窗口中单击上方的“XML”选项按钮，可以查看该配置文件的 xml 格式的配置代码，刚才所做的两项设置，实际上得到了如下的配置段：

```
1：<error-page>
2：        <exception-type>java.lang.Exception</exception-type>
3：        <location>/err.jsp</location>
4：</error-page>
5：<error-page>
6：        <exception-type>404</exception-type>
7：        <location>/building.jsp</location>
8：</error-page>
```

因此，也可直接在 web.xml 文件中修改配置。

11.4　手动发布网站

如果需要手动发布一个网站，只需要将网站的整个目录复制到 Tomcat 的 webapps 目录下，但是网站本身的目录结构必须符合一定的要求。

例如，网站目录名称为 myproject，则其中的 WEB-INF 目录中应存放编译过的 Java 类文

件，若使用的是单个的类，则应将其编译后的.class 文件连同与包层次对应的子目录结构放在 WEB-INF\classes 下面。例如，类文件 MyClass.java 的包名是 com.sunny.test，则应该将 MyClass.class 存放在 myproject\WEB-INF\classes\com\sunny\test 下。

若多个.class 文件已经打包成.jar 文件，则应该将该.jar 文件存放在 WEB-INF\lib 目录下。

网站本身的配置文件 web.xml 也存放在 WEB-INF 目录下。

以上面的网站 myproject 为例，若将其复制到 Tomcat 的 webapps 目录下，则在浏览器中的访问路径是“http://localhost:8080/myproject/文件名”。

11.5　利用 NetBeans 发布网站

小型的网站可以直接手动进行发布，若是大型的网站，则比较困难。我们可以借助 IDE 开发工具来完成网站的发布。

在 MyEclipse 开发环境中，在测试一个页面前就需要将其发布到服务器中，具体操作可参考本书第 20 章的内容。

下面介绍如何利用 NetBeans IDE 来发布网站。

在 NetBeans 的工具栏中，可以看到两个锤子形状的图标，单击它们中的任意一个都可以为主项目生成发布包。二者的区别是，后者先清除在此之前所生成的所有内容，然后重新编译所有页面，重新布署，生成发布包；而前者则仅根据在上次生成之后所做的修改进行增量布署。

如果一个项目不是主项目，则可通过在项目资源窗口中右击项目名称，在弹出的右键菜单中选择“生成”或“清理并生成”，也同样可生成项目的发布包。

完成上述操作后，项目文件夹中将产生新的文件夹 build 和 dist，build 文件夹中存放的是在 NetBeans 中运行网站所需的中间文件，而在 dist 文件夹中只有一个后缀为.war 的文件，它是 Web 项目布署内容的压缩包，也就是我们需要的发布包，只需将该文件复制到 Tomcat 安装目录下面的 webapps 中，即完成了网站的发布。

以前面的 jspbook 项目为例，可在其 dist 目录下找到 jspbook.war，如图 11-11 所示。在 NetBeans 的文件窗口中可以展开 jspbook.war 文件名前面的“+”按钮，打开该压缩文件，可查看其中的内容。

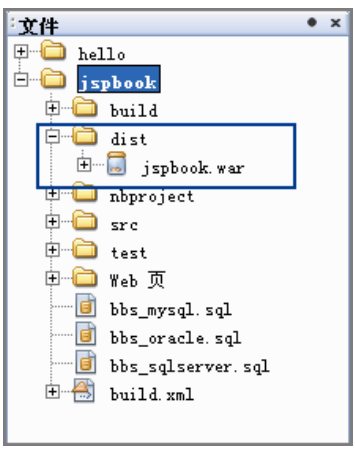


图 11-11　生成的项目发布包

将该文件复制到 Tomcat 安装目录下的 webapps 中，如图 11-12 所示。在复制文件前最好先停止相应的 Tomcat 服务器。

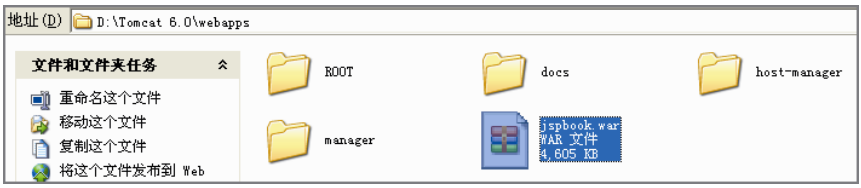


图 11-12　将发布包复制到服务器

然后启动 Tomcat，可以看到 webapps 目录中已经自动添加了一个 jspbook 目录，该目录是由 Tomcat 将 jspbook.war 自动解压而得到的，如图 11-13 所示。

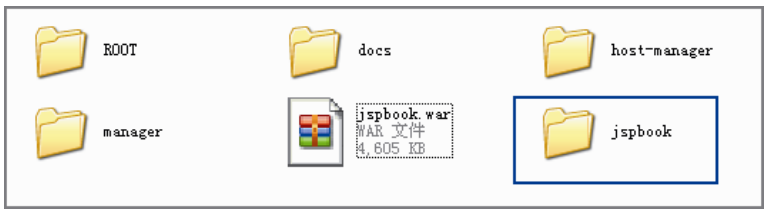


图 11-13　发布包被自动解压

在浏览器中输入 http://localhost:8080/jspbook/t.jsp 访问 t.jsp，得到如图 11-14 所示的运行效果。



图 11-14　发布后的访问结果



思考与训练

一、思考题

1．利用 NetBeans 制作的发布包在项目的什么目录下？如何生成项目的发布包？

2．如何为 JSP 页面单独设置异常处理页面？如何为整个应用配置异常处理方案？

二、综合实训

1．请安装并配置 JSP 的服务器运行环境。

2．请将项目发布到 Tomcat 下，并尝试访问其中的页面。

第三部分　综合案例——企业网站

软件的开发过程不仅仅是技术的简单应用，一般需要经过需求分析、系统设计、模块设计，以及编码、测试、布署等几个阶段。

本部分将详细介绍一个企业网站的开发过程，该网站采用了 MVC 模式实现，同时也通过该项目介绍了一些新的 JSP 技术，如表达式语言 EL、JSTL 标准标签库、过滤器的使用等。希望通过本部分的学习，读者在提高技术能力的同时，也能了解如 MVC 模式等程序设计的思想，从而从多方面提高自身的 IT 素养。具体章节如下。

第 12 章　需求分析
第 13 章　系统设计
第 14 章　数据库设计
第 15 章　功能模块的设计与实现

第 12 章　需 求 分 析

12.1　开 发 背 景

公司门户网站在商业贸易活动中的作用日益明显，越来越多的人利用便捷的因特网查询信息，通过这些信息指导他们进行业务活动，因此公司网站对于扩大公司的影响、提升公司的品牌形象，以及拓展业务都有着非常重要的作用。

本网站的委托开发方是一家成立两年的投资担保公司，对于一个新兴的公司来说，一个展现公司的平台是非常重要的，而网络可以为这个平台提供一个快捷方便的通道。

该公司的原有网站是完全静态的，每一次内容的更新对于网站管理人员来说都是一次“痛苦的折磨”，因为修改网站的工作量非常巨大。同时，旧版网站的主页面过于单调，色调较沉重。因此公司决定对网站进行改版，希望能通过快捷又简便的方式对网站的内容进行更新和维护，为网站设计颇具吸引力的首页，丰富网站的内容，为客户提供及时有效的信息，加强与客户的沟通，以期达到宣传和拓展公司业务的目的。

12.2　需 求 说 明

通过与客户公司相关负责人的沟通，整理出以下基本需求。

12.2.1　用户角色

公司网站的用户类型主要有两种，分别对应不同的权限设置。

1．管理员

该类用户主要为内部的管理人员，该类用户在通过登录验证后，可对网站系统进行维护、设置、发布信息等操作。

系统还应设一个超级管理员。超级管理员具有对普通管理员进行管理的权限。

2．普通用户

无须登录，可浏览企业网站的所有公开的信息页面和下载有关文档。

12.2.2　系统功能概述

根据用户类型的不同，可将系统功能划分为一般用户的前台功能和管理员的后台功能。

1．前台功能

供企业的客户或其他普通用户使用，具体包括以下内容。

1）信息浏览

可浏览网站首页、公司介绍、业务介绍、新闻资讯、政策法规、人才招聘等方面的信息。所设的栏目可分为 4 组，每组的栏目如下。

■ 第 1 组：关于公司。

包含公司简介、经营理念、业务体系、品牌经营、组织架构、企业文化、人力资源、司徽释义等栏目，每个栏目下仅显示置顶的一篇文章的内容。

■ 第 2 组：业务服务。

含企业融资担保、个人贷款担保、经济合同履约担保、担保配套服务等栏目。每个栏目下仅显示置顶的一篇文章的内容。

■ 第 3 组：文章管理。

含新闻速递、政策法规、担保信息、招贤纳士 4 个栏目。每个栏目需要显示多篇文章。

■ 第 4 组：联系咨询。

包括联系我们与在线留言两项功能。

2）表格下载

普通用户可下载所发布的表格资料。这些表格文件由管理员通过后台系统上传。

3）在线留言

普通用户可以给企业留言，这些留言将由管理员进行查看解答。

2．后台功能

管理员对系统的管理，主要包括各栏目信息的发布及管理、设置网站本身的基本信息，以及处理留言。

综上所述，可绘制如图 12-1 所示的系统用例图。

系统的超级管理员还具有对管理员用户进行管理的权限。

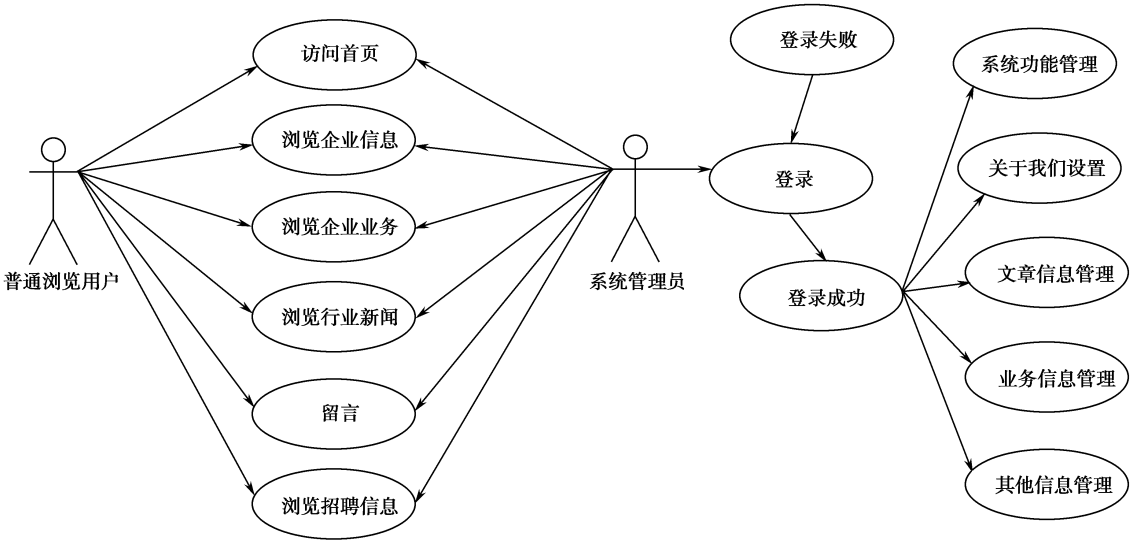


图 12-1　系统用例图

12.3　可行性分析

通过对客户需求的调查与整理，系统的需求已经非常明确，结合对开发人员现有知识结构的了解，无论从技术上、时间上，还是从经济效益的角度进行分析，我们认为该公司网站项目的开发是完全可行的。接下来，我们将根据用户的需求进行系统分析和设计。

第 13 章　系统设计

13.1　系统功能结构

13.1.1　网站的页面级别关系

根据用户浏览网站的一般顺序，可将网页分为不同的级别，如主页是用户首先访问的页面，即一级页面。通过一级页面中的超链接，可以直接跳转到某些包含信息内容的页面，也可能出现的是列表式的二级页面（如新闻标题列表），单击列表中的链接后才会转去相应的内容页面。

各级页面间的关系大致如图 13-1 所示。

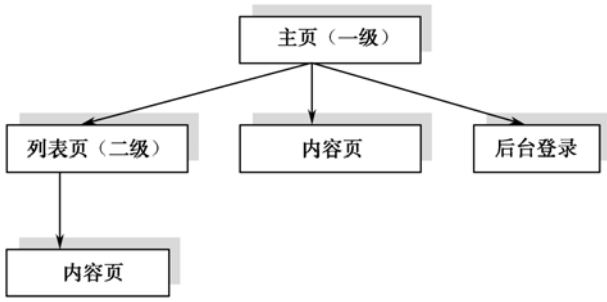


图 13-1　网站页面级别关系图

13.1.2　网站的功能结构

1. 前台功能（见表 13-1）

表 13-1　前台功能一览

首　　页	关于我们	业务服务	新闻速递	政策法规	担保信息	招贤纳士	联系我们
	公司简介 经营理念 业务体系 品牌经营 组织架构 企业文化 人力资源 司徽释义	企业融资担保 个人贷款担保 经济合同履约担保 担保配套服务 业务流程 业务品种 表格下载					在线留言 联系我们
【说明】 无下画线：单页面（布局相同，其中图片样式可不同） 虚线：特定页面 实线：列表页，凡以上列表页的具体信息均由管理员在后台发布管理							

1）前台首页

在首页主体位置显示“新闻速递”、“政策法规”及“担保信息”3 个栏目的最新消息，并在页面上方提供网站导航条，该导航条各链接功能如下。

（1）关于我们：单击“关于我们”超链接后，将在页面左侧出现本组的栏目导航链接，本组别下的这些栏目设置基本是固定不变的，包括公司简介、经营理念、业务体系、品牌经营等。在导航菜单中选择相应栏目后，可以直接查看该栏目下的文章信息。

- 显示方式：以图文结合方式显示。
- 显示内容：公司的各方面信息，介绍公司，展示公司企业文化，经营理念等信息。

（2）业务服务：单击“业务服务”超链接后，在页面左侧出现本组别下的栏目列表，这些栏目也是固定不变的，选择相应的栏目，可以查看公司的相关业务介绍。除表格下载外，每一个栏目下只显示一篇文章的信息。因此单击栏目名称后可直接显示置顶文章的内容。

- 显示方式：以图文结合方式显示。
- 显示内容：展示公司的各项业务。

（3）新闻速递、政策法规、担保信息、招贤纳士：这 4 个栏目每个栏目下都有多篇文章，因此将首先以列表的形式展示文章标题等概况信息，单击链接后进入相关文章的内容页面。

- 显示方式：以列表形式展示全部文章的标题，单击标题后进入具体内容页面，以图文结合方式显示文章的内容。
- 显示内容：列表页显示文章的标题及发布时间，内容详情页则显示文章的标题、内容及发布时间。

（4）联系我们：单击进入联系信息页面，显示公司联系方式信息和留言页面链接。

- 显示方式：单页图文结合方式显示公司联系信息。
- 显示内容：公司联系信息，该信息可通过后台管理进行更新。

2）在线留言

（1）留言内容：联系姓名、电子邮箱（选填）、联系电话（选填）、其他内容。

（2）留言成功：提示留言成功，可返回留言板。

管理员可在后台查看这些留言信息。

2. 后台功能（见表 13-2）

表 13-2　后台功能一览

首　页	系统功能管理	公司信息设置	文章信息管理	业务信息管理	其他信息管理
登录 页面	网站信息查看 管理员设置 网站信息设置 管理员密码修改	公司简介 经营理念 业务体系 品牌经营 组织架构 企业文化	新闻速递 政策法规 担保信息 人才招聘	企业融资担保 个人贷款担保 合同履约担保	表格信息管理 留言板管理 友情链接管理
说明： 无下画线：同一页面（布局相同，其中图片样式可不同） 虚线：特定页面，为满足特殊需求定制 实线：非页面，功能菜单的一级链接，展开的二级链接为其下单元格的内容					

1）系统功能管理

提供网站信息查看、管理员设置、网站基本信息设置和管理员密码修改功能。

（1）网站信息查看：可查看程序路径、Web 服务器信息、MySQl 版本、最大上传限制、数据库容量占用、服务器时间等信息。

（2）管理员设置：可新增、删除管理员，也可修改管理员的密码。

（3）网站基本信息设置：网站名称、网站电话、网站地址、网站邮箱、网站版权。

（4）修改管理密码：输入正确的原始密码后，可修改为新的密码。

2）公司信息的设置

提供公司简介、经营理念、业务体系、品牌经营、组织架构、企业文化等信息的设置。

3）文章信息管理

提供新闻速递、政策法规、担保信息、招贤纳士等文章信息的管理。

（1）查看某一类别的文章标题、发布时间。

（2）添加文章（单页）。添加内容：标题框、内容框、类别（下拉框）。

（3）修改文章：标题框、内容框、类别（下拉框）。

（4）删除文章：删除选定的文章，可多篇同时删除。

4）业务信息管理

提供企业融资担保，个人贷款担保，合同履约担保等信息的发布、删除或修改等管理。

5）其他信息管理

提供表格信息发布、修改或删除，留言板信息查看或删除，以及友情链接的管理。

以上所有栏目的信息管理功能中，除文章信息管理模块中的新闻速递等 4 个栏目外，其余栏目还提供文章置顶功能，即该栏目所属的文章中，设置当前要显示的那一篇文章为置顶文章。

13.1.3　系统架构

本系统采用 MVC（Model View Controller）模式实现。这种模式下，Model 层（即模型层）用于实现系统中的业务逻辑，封装了系统中的业务规则及数据访问处理；View 层（即表现层）实现与用户的交互，用以获取用户的输入信息或向用户显示数据；Controller 层（即控制层）是 Model 层与 View 层之间的桥梁，可根据用户的输入决定调用 Model 层的哪些组件完成业务处理，可分派用户的请求并选择恰当的视图以显示数据。MVC 模式中各层的作用可参见图 13-2。

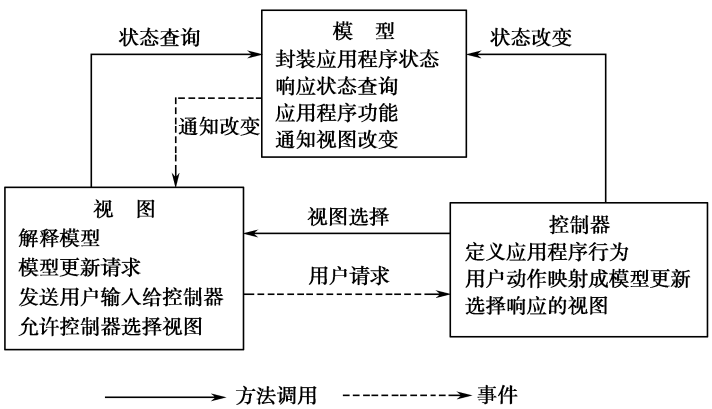


图 13-2　MVC 模式

13.2　开发工具及运行环境

13.2.1　使用技术

功能实现：使用 JSP+Servlet+JavaBean 实现 MVC 模式。
页面展示：使用 DIV+CSS 编写，提高页面打开速度和优化搜索。
在线编辑器：使用 FCKeditor，为后台管理提供丰富的文本编辑功能。

13.2.2　开发工具

IDE 工具：MyEclipse6.0。
该工具的安装及使用方法可参考本书第 20 章的内容。

13.2.3　运行环境

编译器：JDK1.6。
服务器：Tomcat 6.x。
数据库：MySQL。

13.3　项目的文件夹组织结构

将在 MyEclispe 中创建 Web 项目，根据以上的分析，该项目的文件结构设计如图 13-3 和图 13-4 所示。



图 13-3　src 目录的包资源结构

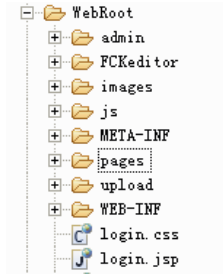


图 13-4　WebRoot 目录的文件结构

src 目录中各包的内容如下。

- com.dao：DAO 接口。
- com.dao.impl：DAO 实现类。
- com.dto：DTO 类。
- com.service：各种 Service 服务类。
- com.servlet：Servlet 类。
- com.util：公共类、工具类。

WebRoot 目录中存放的是各种 JSP 文件、JS 文件、CSS 文件或其他辅助文件，各目录中存放的内容如下。

- admin：后台页面及 CSS、JS 等其他文件。

- pages：前台页面及 CSS、JS 等其他文件。
- images：图片文件。
- js：JavaScript 文件。
- FCKeditor：在线编辑器组件文件。
- upload：上传文件存放目录。

13.4　系 统 预 览

下面展示主要页面的运行效果，以帮助大家进一步理解网站各部分应具有的功能。

13.4.1　前台页面

普通用户访问网站时，首先访问的是网站的前台首页，效果如图 13-5 所示。



图 13-5　网站首页

单击“新闻速递”链接中的新闻标题，可以查看新闻的详细内容，如图 13-6 所示。



图 13-6　新闻内容页面

单击图 13-5 中“新闻速递”后面的“More”链接，将列出所有的新闻标题列表，如图 13-7 所示。



图 13-7　新闻列表页面

栏目“政策法规”、“担保信息”、“招贤纳士”下的文章展示效果与“新闻速递”栏目相同，在此略过。

其他栏目下只有一篇有效文章，将采取直接显示的方式。单击“关于我们”链接，将直接显示该组别下第一个栏目的置顶文章即“公司简介”，效果如图 13-8 所示。其余栏目下的文章显示效果与此相同，这里不再一一展示。



图 13-8　单篇文章展示

单击导航条中的“联系我们”链接，选择“在线留言”，将转到留言页面，如图 13-9 所示，访客可在此输入对企业的留言信息。



图 13-9　用户留言板

13.4.2　后台页面

后台的所有功能需在登录后才能进行，登录页面效果如图 13-10 所示。



图 13-10　后台管理登录页面

登录成功后，可查看网站的基本信息，如图 13-11 所示。

在管理页面左侧菜单栏中选择“管理员设置”，可对管理员用户进行管理，该功能仅对超级管理员开放，普通管理员不具有此项功能，用户管理页面运行效果如图 13-12 所示。

无论是什么级别的管理员，都可修改自己的密码，前提是输入了正确的旧密码。密码修改页面运行效果如图 13-13 所示。

在后台可对各栏目的文章进行增、删、查、改操作，图 13-14 展示的是新闻栏目的管理页面。



图 13-11　后台系统信息管理页面



图 13-12　后台用户管理页面



图 13-13　后台用户密码修改页面



图 13-14　新闻栏目的管理页面

管理员可在各栏目下发布文章，图 13-15 展示的是“新闻速递”栏目下的文章发布页面。

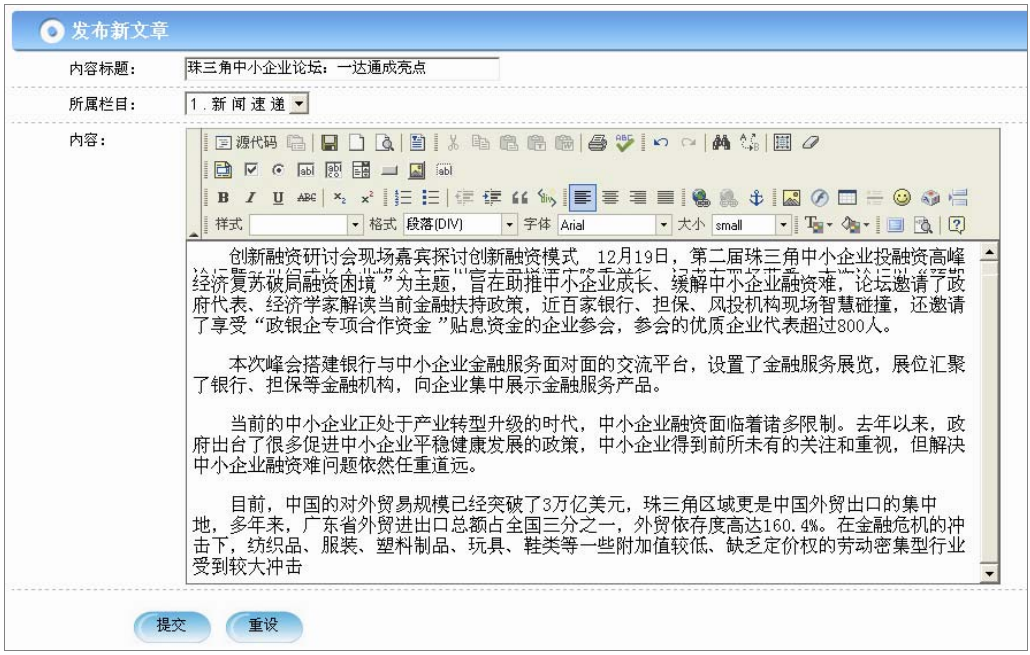


图 13-15　后台文章发布页面

通过单击页面左侧的“其他信息管理”，可在展开的子菜单中选择“留言板管理”，管理页面如图 13-16 所示。

在留言板管理页面中单击每行留言后面的“查看”链接，可以查看留言的详细内容，如图 13-17 所示；单击“删除”链接，则可删除当前行的留言。

可以只列出未查看的留言，如图 13-18 所示，在此页面中通过单击每行后面的“已阅”链接，可以将留言的状态设为“已查看”。



图 13-16　后台留言板管理页面



图 13-17　留言内容查看



图 13-18　查看未查看的留言

第 14 章　数据库设计

经过分析，需创建以下数据表。

（1）t_admin：管理员用户信息表，如表 14-1 所示。

表 14-1　管理员用户信息表结构

表　名	t_admin		名　称		管理员信息表
描述	存储具有后台操作权限的管理员的信息				
索引	admin_id 为主键				
字段	类型	描述	是否为空	默认	备注
admin_id	Int	管理员 ID	Not　null		唯一
admin_name	Varchar（20）	账号	Not　null		
admin_pwd	Varchar（100）	密码	Not　null		
admin_role	Varchar（4）	属性	Null		0：超级管理员 1：普通管理员

（2）t_system：系统信息表，如表 14-2 所示。

表 14-2　系统信息表结构

表　名	t_system		名　称		系统信息表，网站信息表
描述	后台系统的基本信息设置				
索引	sys_id 为主键				
字段	类型	描述	是否为空	默认	备注
sys_id	Int	系统 ID	Not　null		
sys_name	Varchar（50）	网站名称	null		
sys_add	Varchar（80）	网站地址	Null		
sys_mail	Varchar（30）	网站邮箱	Null		
sys_tel	Varchar（30）	联系方式	Null		
sys_copyright	Varchar（150）	版权	Null		

（3）article：文章信息表，如表 14-3 所示。

表 14-3　文章信息表结构

表　名	article		名　称		各栏目的文章信息表
描述	存储各栏目的文章信息				
索引	art_id 为主键				
字段	类型	描述	是否为空	默认	备注
art_id	Int	文章 ID	Not　null		自增

续表

表　名	article		名　称	各栏目的文章信息表	
描述	存储各栏目的文章信息				
索引	art_id 为主键				
字段	类型	描述	是否为空	默认	备注
art_title	Varchar（50）	标题	Not　null		
art_sort	Varchar（10）	类型	Not　null		
art_sortname	Varchar（50）	类型名字	Not　null		
art_content	Varchar（7000）	内容	Not　null		
art_IssueTime	datetime	发表时间	Not　null		
art_group_id	Varchar（2）	栏目所属组别	Not　null		

（4）friendly：友情链接信息表（合作银行），如表 14-4 所示。

表 14-4　友情链接信息表（合作银行）结构

表　名	friendly		名　称	友情链接信息表（合作银行）	
描述	存储本网站的友情链接信息，主要是各合作银行的链接信息				
索引	friendly_id 为主键				
字段	类型	描述	是否为空	默认	备注
friendly_id	Int	友情 ID	Not　null		
friendly_name	Varchar（30）	标题	Not　null		
friendly_imgurl	Varchar（200）	图片路径	Not　null		
friendly_url	Varchar（200）	网址路径	Not　null		

（5）download：表格下载信息表，如表 14-5 所示。

表 14-5　表格下载信息表结构

表　名	download		名　称	表格下载信息表	
描述	供下载的各资料表格的信息				
索引	down_id 为主键				
字段	类型	描述	是否为空	默认	备注
down_id	Int	表格 ID	Not　null		
down_name	Varchar（50）	标题	Not　null		经过系统处理的名称（时间）
down_realname	Varchar（50）	类型	Not　null		原始名称
down_title	Varchar（50）	类型名字	Not　null		显示的名称
down_IssueTime	datetime	时间	Not　null		

（6）message：留言信息表，如表 14-6 所示。

表 14-6　留言信息表结构

表　名	message		名　称		留言信息表
描述	客户所发布的留言信息及状态信息				
索引	massage_id 为主键				
字段	类型	描述	是否为空	默认	备注
msg_id	Int	留言版 ID	Not　null		
msg_company	Varchar（20）	公司	Not　null		
msg_industry	Varchar（20）	所属行业	Not　null		
msg_area	Varchar（20）	经营范围	Not　null		
msg_time	Varchar（20）	成立时间	Not　null		
msg_name	Varchar（10）	联系人	Not　null		
msg_phone	Varchar（20）	电话	Not　null		
msg_fax	Varchar（20）	传真	null		
msg_mail	Varchar（50）	邮箱	Not　null		
msg_content	Varchar（2000）	留言内容	null		
issueTime	datetime	发表时间	Not null		
msg_role	char（1）	留言的查看状态	Not　null	0	0 为未查看 1 为已查看

第 15 章　功能模块的设计与实现

15.1　公　共　类

在系统的开发过程中，有些功能是可被许多模块所共享的，为了在开发的过程中最大限度地实现代码重用，加快项目的开发速度，可将这些通用的功能封装成公共类，将来直接调用即可，下面列举本系统中主要公共类的实现。

15.1.1　数据库连接类

动态网站开发中需要频繁地完成数据库连接与断开等操作，这里设计的数据库连接类可以根据属性文件中的参数设置，决定采用数据库连接池还是直接使用 JDBC 获取数据库连接，该类的主要方法如下。

- config：读取属性文件 database.properties 中的数据库连接参数。
- getConnection：获取数据库连接池或直接使用 JDBC 获取数据库连接。
- releaseConnection：释放（连接池模式）或关闭（JDBC）数据库连接。
- commit：提交事务。
- rollback：事务回滚。

该类具体代码如下所示：

程序清单　src\com\util\Database.java

```
1：package com.util;
2：
3：import java.sql.Connection;
4：import java.sql.DriverManager;
5：import java.sql.SQLException;
6：import java.util.Enumeration;
7：import java.util.Hashtable;
8：import java.util.PropertyResourceBundle;
9：import javax.naming.Context;
10：import javax.naming.InitialContext;
11：import javax.naming.NamingException;
12：import javax.sql.DataSource;
13：
14：public class Database {
15：  private static String url;
16：  private static String driver;
17：  private static String username;
18：  private static String password;
19：  private static String type;
20：  private static String datasource;
```

```
21:  private final static String fileName = "database";
22:  private static ThreadLocal connection = new ThreadLocal();
23:  static {
24:          config();
25:  }
26:
27: private static void config() {
28:  PropertyResourceBundle resourceBundle = (PropertyResourceBundle)
              PropertyResourceBundle.getBundle(fileName);
29:  Enumeration enu = resourceBundle.getKeys();
30:  while (enu.hasMoreElements()) {
31:          String propertyName = enu.nextElement().toString();
32:          if (propertyName.equals("database.url"))
33:                  url = resourceBundle.getString("database.url");
34:          if (propertyName.equals("database.driver"))
35:                  driver = resourceBundle.getString("database.driver");
36:          if (propertyName.equals("database.username"))
37:                  username = resourceBundle.getString("database.username");
38:          if (propertyName.equals("database.password"))
39:          password = resourceBundle.getString("database.password");
40:          if (propertyName.equals("database.type"))
41:                  type = resourceBundle.getString("database.type");
42:          if (propertyName.equals("database.datasource"))
43:  datasource = resourceBundle.getString("database.datasource");
44:  }
45: }
46:
47: public synchronized static java.sql.Connection getConnection()
48:          throws SQLException {
49:  Connection con = (Connection) connection.get();
50:  if (con != null && !con.isClosed()) {
51:          return con;
52:  }
53:  if ("pooled".equalsIgnoreCase(type)) {
54:  try {
55:          Hashtable env = new Hashtable();
56:          Context ctx = new InitialContext(env);
57:                  DataSource dataSource = (DataSource) ctx.lookup(datasource);
58:                          con = dataSource.getConnection();
59:                          connection.set(con);
60:                          return con;
61:                  } catch (NamingException e) {
62:                          e.printStackTrace();
63:          }
64:  } else {
65:  // 直接采用 JDBC 获取连接
```

```
66:  try {
67:          Class providerClass = Class.forName(driver);
68:          con = DriverManager.getConnection(url, username, password);
69:                con.setAutoCommit(false);
70:                connection.set(con);
71:                return con;
72:
73:          } catch (ClassNotFoundException e) {
74:                e.printStackTrace();
75:          }
76:  }
77:  return null;
78: }
79:
80: public static void commit() {
81:  Connection con = (Connection) connection.get();
82:  try {
83:          con.commit();
84:  } catch (SQLException e) {
85:          e.printStackTrace();
86:  }
87: }
88:
89: public static void rollback() {
90:  Connection con = (Connection) connection.get();
91:  try {
92:          con.rollback();
93:  } catch (SQLException e) {
94:          e.printStackTrace();
95:  }
96: }
97:
98: public synchronized static void releaseConnection(Connection connection)
99: {
100:         try {
101:               if (connection != null && !connection.isClosed())
102:                     connection.close();
103:         } catch (SQLException e) {
104:               e.printStackTrace();
105:         }
106:         connection = null;
107: }
108: }
```

15.1.2　通用处理工具类

该类主要包含了对字符串所做的一些通用处理，主要方法如下。

- convertStringNull：将 null 字符串转换为空串。
- toChineseStr：将字符串转换为按 UTF-8 解码的字符串。
- getDateTimeStr：获取格式化的当前时间字符串。
- getPageLinkStr：根据记录总数、当前页号、每页显示的记录条数生成页号导航链接字符串。
- encodeHTML：将字符串中的 HTML 标签及特殊符号转换为转义字符。
- decodeHTML：是 getHTML 方法的逆向操作，将转义字符还原为 HTML 标签及符号。
- checkSqlInject：检测 SQL 语句字符串中是否包含有可能导致 SQL 注入问题的敏感字符串，是则返回 true，否则返回 false。
- encodeLoginStr：将登录信息中的 HTML 特殊符号转换为转义字符。

下面仅列出该类中的主要方法，更多的方法请参考本书所附源码：

主要代码　src\com\util\ToolMethod.java

```
1：package com.util;
2：
3：import java.text.SimpleDateFormat;
4：import java.util.Date;
5：
6：public class ToolMethod {
7：     public static final String convertStringNull(String strOrig)
8：       {
9：            String strReturn = "";
10：           if (strOrig == null || strOrig.equals(null)){
11：                strReturn = "";
12：           } else{
13：                strReturn = trim(strOrig.trim());
14：           }
15：           return strReturn;
16：      }
17：//将字符串转换为 UTF-8 编码
18：public static final String toChineseStr(String strOrigin) {
19：     if (strOrigin == null || strOrigin.equals(null)){
20：          strOrigin = "";
21：      }else{
22：          strOrigin = strOrigin.trim();
23：      }
24：      try{
25：          strOrigin = new String(strOrigin.getBytes("ISO8859_1"), "UTF-8");
26：          }catch (Exception e){    }
27：          return strOrigin;
28：  }
```

```
29：
30：public static final String getDateTimeStr() {
31：  SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
32：  String time = f.format(new Date());
33：  return time;
34：}
35：
36：// count 记录总数，perpage 每页显示的条数，curPage 当前页号
37：public static final String getPageLinkStr(int Count, int perPage,
 int curPage,      String url) {
38：        if (Count <= perPage)
39：              return null;
40：        StringBuffer buff = new StringBuffer();
41：        double pageDou = (double) Count / (double) perPage;
42：        int pageInt = Count / perPage;
43：        int pageCount;
44：        if (pageDou > pageInt)
45：              pageCount = pageInt + 1;
46：        else
47：              pageCount = pageInt;
48：        buff.append("共" + pageCount + "页    ");
49：        buff.append("当前页：" + curPage + "    ");
50：        if (curPage > 1)
51：              buff.append("<a href=\"" + url + (curPage - 1)
52：                          + "\">上一页</a>   ");
53：        else
54：              buff.append("");
55：        if (curPage < pageCount)
56：              buff.append("<a href=\"" + url + (curPage + 1)
57：                          + "\">下一页</a>    ");
58：        else
59：              buff.append("");
60：        buff.append("第\n");
61：        buff.append("<select onchange=\"test(this)\" name=\"select\">
                \n");
62：        for (int i = 1; i <= pageCount; i++) {
63：              if (i == curPage)
64：                    buff.append("<option selected>" + i + "</option>\n");
65：              else
66：                    buff.append("<option>" + i + "</option>\n");
67：        }
68：        buff.append("</select>\n");
69：        buff.append("页");
70：        buff.append("<script language=\"JavaScript\">\n");
71：        buff.append("function test(page){\n"};
72：        buff.append("window.location=\"" + url
```

```
73：                        + "\"+page.options[page.selectedIndex].text;\n");
74：        buff.append("</script>\n");
75：        return buff.toString();
76：  }
77：  public static final String encodeHTML(String input) {
78：        if (input == null) {
79：              input = "";
80：              return input;
81：        }
82：        input = input.trim().replaceAll("&", "&amp;");
83：        input = input.trim().replaceAll("<", "&lt;");
84：        input = input.trim().replaceAll(">", "&gt;");
85：        input = input.trim().replaceAll("\t", "        ");
86：        input = input.trim().replaceAll("\r\n", "\n");
87：        input = input.trim().replaceAll("\n", "<br>");
88：        input = input.trim().replaceAll("    ", "  ");
89：        input = input.trim().replaceAll("'", "&#39;");
90：        input = input.trim().replaceAll("\\\\", "&#92;");
91：        return input;
92：  }
93：  // 检测 SQL 语句是否有注入危险
94：  public static boolean checkSqlInject(String str) {
95：// SQL 字符串中不许出现的单词
96：        String inj_str = "'|and|exec|insert|select|delete|update|count
            |*|%|chr|mid|master|truncate|char|declare|;|or|-|+|,";
97：        String[] inj_stra = inj_str.split("\\|");
98：        for (int i = 0; i < inj_stra.length; i++) {
99：              if (str.indexOf(inj_stra[i]) >= 0) {
100：                        return true;
101：                  }
102：              }
103：              return false;
104：        }
105：  }
```

15.1.3　编码转换过滤器

在中文网站的开发中，来自客户端的请求参数可能包含中文内容，而发往客户端的响应内容一般也包含有中文内容，为应用配置一个过滤器，将大大简化对请求或响应的编码转换处理。该过滤器的代码如下：

程序清单　src\com\util\CharacterEncodingFilter.java

```
1：package com.util;
2：
3：import java.io.IOException;
4：import javax.servlet.Filter;
```

```
5：import javax.servlet.FilterChain;
6：import javax.servlet.FilterConfig;
7：import javax.servlet.ServletException;
8：import javax.servlet.ServletRequest;
9：import javax.servlet.ServletResponse;
10：
11：public class CharacterEncodingFilter implements Filter {
12：  private String encoding = null;
13：  private FilterConfig filterConfig = null;
14：  private boolean ignore = true;
15：
16：  public void destroy() {
17：        encoding = null;
18：        filterConfig = null;
19：  }
20：
21：public void doFilter(ServletRequest request, ServletResponse response,
      FilterChain filterChain) throws IOException, ServletException {
22：        if (ignore == true || request.getCharacterEncoding() == null) {
23：              if (encoding != null) {
24：                    request.setCharacterEncoding(encoding);
25：                        response.setCharacterEncoding(encoding);
26：              }
27：        }
28：        filterChain.doFilter(request, response);
29：}
30：
31：public void init(FilterConfig filterConfig) throws ServletException {
32：  this.filterConfig = filterConfig;
33：  this.encoding = filterConfig.getInitParameter("encoding");
34：  String value = filterConfig.getInitParameter("ignore");
35：  this.ignore= value == null||value.equalsIgnoreCase("true");
36：   }
37：}
```

当此过滤器未设置参数 ignore 的值或 ignore 的值设为 true 时，将采用过滤器参数 encoding 的值作为默认的编码字符集。要使过滤器生效，需在配置文件 web.xml 中进行正确的配置，相关配置段如下：

```
1：<filter>
2：      <filter-name>CharacterEncodingFilter</filter-name>
3：      <filter-class>com.util.CharacterEncodingFilter</filter-class>
4：      <init-param>
5：         <param-name>encoding</param-name>
6：         <param-value>UTF-8</param-value>
7：      </init-param>
8：</filter>
```

```
9：<filter-mapping>
10：        <filter-name>CharacterEncodingFilter</filter-name>
11：        <url-pattern>/*</url-pattern>
12：</filter-mapping>
```

上面的配置段中，第 4～7 行设置了参数 encoding 的值（即默认采用的编码字符集）为 UTF-8，未设置参数 ignore 的值。

15.1.4　数据库通用处理

在对数据库各表所进行的操作中，有的处理过程是基本相似的，可将它们封装成公共的接口和类，作为其他各 DAO 的父接口和父类。

1．通用操作接口 BaseDAO

程序清单 src\com\dao\BaseDAO.java

```
1：package com.dao;
2：
3：import java.sql.Connection;
4：import java.sql.SQLException;
5：
6：public interface BaseDao {
7：/* id: 要删除的 ID 值列表(数组表示)
8：  * tableName:表名
9：  * keyId:id 列的名称
10：  * 作用：删除表中一批记录，条件：其 ID 列的值在列表中
11：  */
12：  public boolean del(String id[], String tableName, String keyId)
                throws SQLException;
13：
14：/* table:表名
15：  * 作用：返回表中记录的总条数
16：  */
17：  public int getSum(String table)throws SQLException;

18：/* table:表名
19：  * type: 列名
20：  *       ID:键值
21：  * 作用：返回表中符合条件的记录条数 条件：type 列的值与 id 相同
22：  */
23：  public int getSum(String table, String type, String id)
                throws SQLException;
24：
25：//作用：设置操作中所要使用的连接对象
26：  public void setConnection(Connection conn);

27：//作用：返回正在使用的连接对象
28：  public Connection getConnection();
```

```
29：}
```

2．通用操作实现类 BaseDAOImpl

程序清单 src\com\dao\impl\BaseDAOImpl.java

```
1：package com.dao.impl;
2：
3：import java.sql.Connection;
4：import java.sql.PreparedStatement;
5：import java.sql.ResultSet;
6：import java.sql.SQLException;
7：import com.dao.BaseDao;
8：public class BaseDaoImpl implements BaseDao {
9：    Connection connection = null;
10：  public Connection getConnection() {
11：        return connection;
12：  }
13：  public void setConnection(Connection connection) {
14：        this.connection = connection;
15：  }
16：  public boolean del(String[] id, String tableName, String keyId)
    throws SQLException {
17：        StringBuffer sql = new StringBuffer("delete from "+
                                    tableName+" where "+keyId+" in('");
18：        for (int i = 0; i < id.length; i++){
19：              if(i < id.length - 1)
20：                    sql.append(id[i]+"','");
21：              else
22：                    sql.append(id[i]+"')");
23：        }
24：        //执行删除
25：        PreparedStatement ps = connection.prepareStatement(
                                        sql.toString());
26：        ps.executeUpdate();
27：        ps.close();
28：        return true;
29：  }
30：
31：  public int getSum(String table, String type, String id)
                throws SQLException {
32：        String sql = "select count(*) count from " + table +" where "
                    + type + " = " + id;
33：        return getTableSum(sql);
34：  }
35：
36：  public int getSum(String table) throws SQLException {
37：        String sql = "select count(*) count from " + table;
```

```
38：        return getTableSum(sql);
39：  }
40：
41：  public int getTableSum(String sql) throws SQLException {
42：        //构成 SQL 语句
43：        PreparedStatement ps = connection.prepareStatement(sql);
44：        ResultSet rs = ps.executeQuery();
45：        while(rs.next()){
46：              return rs.getInt("count");
47：        }
48：        return 0;
49：  }
50：}
```

3．通用操作服务类 BaseService

该类对 BaseDAOImpl 类提供的操作做了进一步的封装，使得控制层调用时不必理会数据库连接与关闭等细节问题，代码请参考项目中的 src\com\service\BaseService.java 文件。

15.1.5　公共页面

在前台模块中，页面头部的内容基本相同，主要包括网站导航条等信息，可将此部分内容设计成公共页面，其他需要的页面中只需包含此页面即可。

该页面（head.jsp）的代码比较简单，可参考本书所附源码，这里不再列出。

15.2　Model 层的设计

15.2.1　实体类设计

实体类用于封装来自或即将写入数据表中的数据，本系统的数据库结构相对简单，可参照数据库的表结构，设计相应的实体类，类名基本与表名相同，各类的属性名则与表中各列的列名相同。本系统中设计了如下的实体类，均处于包 com.dto 中。

- Admin：封装了管理员信息的实体类。
- SystemInfo：封装了系统信息的实体类。
- Article：封装了文章信息的实体类。
- FriendLink：封装了友情链接信息的实体类。
- Download：封装了下载表格信息的实体类。
- Message：封装了留言板信息的实体类。

15.2.2　DAO 接口

与数据库的相关操作是业务处理的关键，对系统的业务流程进行分析后，可以得到与各数据表相关的数据库操作，与各表相关的操作接口设计如下。

1．AdminDao 接口

对管理员用户信息表的操作，具有以下方法：

```
1：//保存新增的管理员信息
2：public boolean addAdminSave(Admin admin)throws SQLException;
3：
4：//根据 ID 修改管理员的密码
5：public boolean editAdminPassword(String id,Admin admin)
     throws SQLException;
6：
7：//检验用户名是否存在
8：public boolean uniqueNameCheck(Admin admin)throws SQLException;
9：
10：//列出除超级管理员外的所有管理员信息
11：public List listAllAdminExceptSuper(int curPage, int perpage)
   throws SQLException;
12：
13：//根据 ID 获取管理员用户信息
14：public Admin getAdminById(String id) throws SQLException;
15：
16：//根据字段值获取管理员信息
17：public Admin getAdminByField(String fieldName, String fieldValue)
  throws SQLException;
```

2．ArticleDao 接口

对文章信息表的操作，包含的方法如下：

```
1：// 保存新添加的文章信息
2：    public boolean addArtSave(Article art)throws SQLException;
3：
4：// 根据组别和栏目 ID 得到下拉菜单
5：    public List<Article> listArtBySortId(String id,String group_id)
   throws SQLException;
6：
7：// 根据文章 ID 获取文章信息
8：    public Article getArtById(int id)throws SQLException;
9：
10：//       根据 ID 保存文章的修改信息
11：     public boolean editArtSave(String id,Article art)throws SQLException;
12：
13：//根据组别列出所有的栏目名称
14：   public List listSortNameByGroup(String group_id) throws SQLException;
15：
16：   //获取某个一级栏目的文章(下属文章只有一篇)
17：   public Article getFirstArt(String group_id,String sort_id)
                    throws SQLException;
18：
19：   //获取某个栏目指定页的文章列表
20：   public List listArtInOnePage(String group_id,String sort_id,
```

```
        int curPage, int perpage)throws SQLException;
21：
22：  //文章置顶操作
23：public boolean doTop(String[] id) throws SQLException;
```

3．DownloadDao 接口

该接口包含了对表格下载信息表的基本操作，设计如下：

```
1：//添加供下载表格的信息
2：public boolean addFormSave(Download form) throws SQLException;
3：
4：//根据页号列出某页的下载表格的信息
5：public List listFormsInOnePage(int curPage, int perpage)
6：        throws SQLException;
7：
8：//根据 ID 获取某表格的信息
9：public Download getFormById(int id) throws SQLException;
10：
11：//修改表格标题
12：public boolean editFormTitleSave(Download form) throws SQLException;
```

4．FriendLinkDao 接口

该接口包含了对友情链接信息表的操作，设计如下：

```
1：//保存新增的友情链接信息
2：public boolean addFriendLinkSave(FriendLink fri)throws SQLException;
3：
4：//列出指定页的所有友情链接信息
5：public List listFriendLink(int curPage, int perpage)throws SQLException;
6：
7：//根据 ID 获取一条友情链接信息
8：public FriendLink getLinkById(int id)throws SQLException;
9：
10：//保存所修改的友情链接信息
11：public boolean editFriendLinkSave(FriendLink fri)throws SQLException;
12：
13：//对选中的友情链接信息置顶
14：public boolean doTop(String[] id) throws SQLException;
```

5．MessageDao 接口

该接口封装了对留言信息表的基本操作，设计如下：

```
1：// 保存新添加的留言信息
2：public boolean addMsgSave(Message message) throws SQLException;
3：
4：// 根据条件得到留言列表
5：public List listMsg(int curPage, int perpage, String condition)
6：        throws SQLException;
```

```
7：
8：// 根据 ID 得到留言信息
9：public Message getMsgById(int id) throws SQLException;
10：
11：// 修改留言的查看状态为“已查看”
12：public boolean editMsgSave(String id) throws SQLException;
```

6．SystemDao 接口

该接口封装了系统信息表的操作，设计如下：

```
1：//更新系统信息
2：public boolean updateSystemInfo(SystemInfo system)throws SQLException;
3：
4：//读取系统信息，ID 参数预留，一般为 1
5：public SystemInfo getSystemInfoById(String id) throws SQLException;
```

以上所有的接口均存放在包 com.dao 中，每个接口都有对应的实现类，名称为 XXXImpl 的形式，其中“XXX”为接口名称，实现类所在的包为 com.dao.impl。各方法的具体实现将结合各功能模块进行介绍。

15.3　前台功能模块

所有前台页面均保存在 WebRoot\pages 子目录下，前台除留言模块外，主要完成数据的显示功能，其基本处理流程如下。

用户在页面上单击超链接或以其他方式访问网站时，将由相应的 Servlet 接收用户请求，调用相应的 Service 服务类，而 Service 类通过相应的 DAO 实现类完成对数据库数据的读取操作，并将读取出来的数据封装在 DTO 对象中，这些 DTO 对象将作为 request 范围的属性内容转发给展示页面，在页面中通过 EL 和 JSTL 技术获取并显示这些数据。

15.3.1　首页显示模块

首页除提供网站导航的链接外，还需要显示新闻速递、政策法规等栏目的信息。这里以首页新闻栏目的文章显示为例，说明采用 MVC 模式实现数据展示的一般方法。

本模块相关资源如表 15-1 所示。

表 15-1　首页显示模块主要资源

编　号	类名/文件名	相 关 方 法	功 能 描 述	所 处 位 置
1	index.jsp		显示前台首页的内容	WebRoot\pages
2	ShowIndexServlet	doGet doPost	调用 Service 读取“新闻速递”等 3 个栏目的最新文章，以便在首页显示	com.servlet 包
3	ArticleService	listArtPage 作用：获取指定栏目指定页的文章列表	在 ArticleDaoImpl 类的基础上进一步封装，提供对表 article 操作的相应服务	com.service 包

续表

编　号	类名/文件名	相 关 方 法	功 能 描 述	所 处 位 置
4	Article	所有 set/get 方法	封装了文章信息的实体类	com.dto 包
5	ArticleDao（接口）	listArtPage	封装了对表 article 的所有操作	com.dao 包
6	ArticleDaoImpl	listArtPage 作用：获取指定栏目指定页的文章列表	ArticleDao 接口的具体实现	com.dao.impl 包

在本系统中，各类资源的存放位置可参考 13.3 节中的文件夹结构示意，下文不再指出。首先应在 web.xml 中对网站默认首页进行配置，相关配置段如下：

相关配置　WebRoot\WEB-INF\web.xml

```
1：      <servlet>
2：          <servlet-name>ShowIndexServlet</servlet-name>
3：          <servlet-class>com.servlet.ShowIndexServlet</servlet-class>
4：      </servlet>
5：      <servlet-mapping>
6：          <servlet-name>ShowIndexServlet</servlet-name>
7：          <url-pattern>/index.jsp</url-pattern>
8：      </servlet-mapping>
9：
10：     <welcome-file-list>
11：        <welcome-file>index.jsp</welcome-file>
12：    </welcome-file-list>
```

做了以上的设置后，当用户访问网站首页时，系统将自动将请求提交给 ShowIndexServlet 进行处理。下面仅列出本模块各资源的主要代码。

1．ShowIndexServlet 类

主要代码　src\com\servlet\ShowIndexServlet.java

```
1：package com.servlet;
2：//下面略去 import 语句段
3：......
4：public class ShowIndexServlet extends HttpServlet {
5：    public void doGet(HttpServletRequest request,
     HttpServletResponse response) throws ServletException, IOException {
6：        doPost(request,response);
7：    }
8：    public void doPost(HttpServletRequest request,
       HttpServletResponse response) throws ServletException, IOException {
9：        ShowIndex(request,response);
10：   }
11：   private void ShowIndex(HttpServletRequest request,
     HttpServletResponse response) throws ServletException, IOException{
12：        response.setContentType("text/html;charset=utf-8");
```

```
13：        PrintWriter out = response.getWriter();
14：        ArticleService as = new ArticleService();
15：      //获取新闻速递栏目的最新 8 篇文章
16：request.setAttribute("ListXing",as.listArtPage("2","1",1,8));
17：      //获取政策法规栏目的最新 8 篇文章
18：request.setAttribute("ListZheng",as.listArtPage("2","2",1,8));
19：      //获取担保信息栏目的最新 8 篇文章
20：request.setAttribute("ListDan",as.listArtPage("3","1",1,8));
21：//获取合作银行列表
22：FriendLinkService cs = new FriendLinkService();
23：request.setAttribute("ListFriendly",cs.Friendly(1,10));
24：//获取版权信息
25：        SystemService sy = new SystemService();
26：        request.setAttribute("Sy",sy.IdSystem("1"));
27：            RequestDispatcher rd = request.getRequestDispatcher(
                "pages/index.jsp");
28：        rd.forward(request,response);   //请求转发
29：        out.flush();
30：        out.close();
31：  }
32：}
```

2．ArticleService 类的相关代码

该类实际上是对 ArticleDaoImpl 中相关方法的进一步封装，使得调用者不必关心一些操作的细节，如数据库的连接与关闭，异常处理等。其他模块中所涉及的 XXXService 类的方法均与此相似，后面将不再一一列出其代码。

```
相关代码　src\com\service\ArticleService.java
1：package com.service;
2：//略去 import 语句段
3：......
4：public class ArticleService extends BaseService{
5：private ArticleDao ad = new ArticleDaoImpl();
6：//其他方法略
7：......
8：public List listArtPage(String group_id,String sort_id,
                                    int curPage, int perpage){
9：   Connection connection = null;
10：  try {
11：        connection = Database.getConnection();
12：        ad.setConnection(connection);
13：            return ad.listArtPage(group_id,sort_id,curPage,perpage);
14：  } catch (Exception e) {
15：              e.printStackTrace();
16：              Database.rollback();
17：              return null;
```

```
18：          } finally {
19：                Database.releaseConnection(connection);
20：          }
21：     }
22：}
```

3．ArticleDaoImpl 类的相关代码

```
相关代码　com\dao\impl\
1：package com.dao.impl;
2：......
3：public class ArticleDaoImpl extends BaseDaoImpl implements ArticleDao{
4：//其他方法的代码略去。
5：   //根据页号、组别、栏目 ID 查找所有文章列表
6：public List listArtPage(String art_sort,String art_group_id,int curPage,
                          int perpage)throws SQLException{
7：List array = new ArrayList();
8：String sql = "select * from article where art_sort = ?
     and art_group_id=? order by art_top desc limit ?,?";
9：PreparedStatement ps = connection.prepareStatement(sql);
10：ps.setString(1, art_sort);
11：ps.setString(2,art_group_id);
12：ps.setInt(3,(curPage-1)*perpage);
13：ps.setInt(4,perpage);
14：ResultSet rs = ps.executeQuery();
15：while(rs.next()){
16：  Article art = new Article();
17：  art.setArt_group_id(art_group_id);
18：  art.setArt_id(rs.getInt("art_id"));
19：  art.setArt_title(rs.getString("art_title"));
20：  art.setArt_sort(art_sort);   //sort id---栏目 id 号
21：  art.setArt_sortname(rs.getString("art_sortname"));
22：  art.setArt_IssueTime(rs.getString("art_IssueTime").
                                substring(0,10));
23：  art.setArt_content(rs.getString("art_content"));
24：        array.add(art);
25：  }
26：  rs.close();
27：  ps.close();
28：  return array;
29：}
30：}
```

4．实体类 Article 的主要代码

```
主要代码　src\com\dto\Article.java
1：package com.dto;
2：
```

```
3：public class Article{
4：    private int art_id;
5：    private String art_group_id;
6：    private String art_title;
7：    private String art_sort;
8：    private String art_sortname;
9：    private String art_IssueTime;
10：   private String art_content;
11：   private String art_top;
12：   // 生成所有属性的 set/get 方法
13：   .....
14：   }
```

5．前台主页 index.jsp 的主要代码

```
主要代码　WebRoot\pages\index.jsp
1：<body>
2：<div id="background">
3：    <div id="background_2">
4：       <%@include file="head.jsp" %>
5：           <div id="left">
6：          <div id="first">
7：             <ul>
8：                <li>
9：                <a class="li_4" href="ShowArtServlet?
            method=listArt&group_id=1">关于我们</a>
10：               </li>
11：                <li>
12：                 <a class="li_4" href="ShowArtServlet?
                  method=lisArt&group_id=3">业务服务</a>
13：               </li>
14：                <li><a class="li_4" href="ShowArtServlet?
              method=listArt&group_id=2">行业信息</a></li>
15：            </ul>
16：          </div>
17：          <div id="second"><a href="ShowTableServlet?method=List">
18：          <img src="pages/img/download.jpg" /></a>
19：          </div>
20：          <div id="third">
21：            <div id="third_1"></div>
22：            <div id="third_2">
23：            <c:forEach var="link" items="${ListFriendly}">
24：            <a href="${link.friendly_url}" target="_blank">
25：              <img src="pages/htm/${link.friendly_imgurl}"/>
26：              </a>
27：            </c:forEach>
28：            </div>
```

```
29:             <div id="third_3"></div>
30:           </div>
31:         </div>
32:         <!--left end-->
33:         <div id="right">
34:           <div class="content">
35:             <div class="ico">新闻速递</div>
36:             <div class="more">
37:              <a href="ShowArtServlet?method=listArt&group_id=2&sort_id=1">
38:                <img src="pages/img/more.png" />
39:              </a>
40:             </div>
41:             <div class="line"></div>
42:             <div class="title">
43:               <ul>
44:               <c:forEach var="Art" items="${ListXing}">
45:                 <li><span>${Art.art_IssueTime}</span>
46:                  <a href="ShowArtServlet?method=oneArticle
              &art_id=${Art.art_id}" target="_blank">${Art.art_title}
47:                 </a></li>
48:               </c:forEach>
49:               </ul>
50:             </div>
51:           </div>
52:           <div class="content">
53:             <div class="ico">政策法规</div>
54:             <div class="more"><a href="ShowArtServlet?
                    method=listArt&group_id=2&sort_id=2">
55:             <img src="pages/img/more.png" /></a></div>
56:             <div class="line"></div>
57:             <div class="title">
58:               <ul>
59:               <c:forEach var="Art" items="${ListZheng}">
60:               <li><span>${Art.art_IssueTime}</span>
61:                  <a href="ShowArtServlet?method=oneArticle
                        &art_id=${Art.art_id}" target="_blank">
62:                   ${Art.art_title}
63:                  </a>
64:                  </li>
65:               </c:forEach>
66:               </ul>
67:             </div>
68:           </div>
69:           <div class="content">
70:             <div class="ico">担保信息</div>
71:             <div class="more">
72:             <a href="ShowArtServlet?method=listArt&group_id=3&sort_id=3">
```

```
73:                 <img src="pages/img/more.png" />
74:               </a>
75:             </div>
76:             <div class="line"></div>
77:             <div class="title">
78:               <ul>
79:               <c:forEach var="Art" items="${ListDan}">
80:               <li><span>${Art.art_IssueTime}</span>
81:                     <a href="ShowArtServlet?method=oneArticle
                          &art_id=${Art.art_id}" target="_blank">
82:                       ${Art.art_title}
83:                     </a></li>
84:               </c:forEach>
85:               </ul>
86:             </div>
87:           </div>
88:         </div>
89:   </div>
90:     <!--background_2 end-->
91:     <div id="bt">${Sy.sys_copyright}</div>
92: </div>
93: <!--background end-->
94: <div id="background_bt"></div>
95: </body>
```

可以看到，ShowIndexServlet 将需要在页面展示的数据保存在 request 中，而显示页面则通过 JSP 的 EL（表达式语言）读取了这些数据，并结合 JSTL 的标签将数据显示在页面上。

为了节省篇幅，下文中凡涉及显示页面的都只列出数据展示部分的代码。

15.3.2　文章显示模块

每一栏目下可以包含有多篇文章，但展示时则分为两类：一类是仅显示置顶的一篇文章，另一类则是显示栏目下面的所有文章。第二种情况下，应首先显示这些文章的标题列表，当用户单击了文章标题后，再进入文章内容显示页面。而第一种情况下，当用户单击了栏目名称后，将直接显示置顶文章的内容。文章展示模块的主要资源如表 15-2 所示。

表 15-2　文章展示模块主要资源

编　　号	类名/文件名	相 关 方 法	功 能 描 述
1	ShowArtServlet	listArt	显示某个栏目的文章列表或该栏目置顶文章的内容
		showArticleDetail	根据文章 ID 显示某篇文章的内容
2	ArticleService 类 ArticleDaoImpl 类 ArticleDao 接口	listArtPage	获取某一页的文章列表
		getArtById	根据 ID 获取一篇文章
		getFirstArt	获取置顶文章
		listSortNameByGroup	列出某组中所有的栏目信息

续表

编　　号	类名/文件名	相 关 方 法	功 能 描 述
3	Article	所有 set/get 方法	对文章信息的封装，实体类
4	showArticleList.jsp		显示文章列表
5	showArticleDetail.jsp		显示文章内容等详情
6	showArticleDetailDirect.jsp		显示置顶文章的内容
7	contactUs.jsp		显示“联系我们”页面的内容

本网站各栏目的设置基本固定，其中“新闻速递”等 4 个栏目下需要显示文章列表，除此之外，“联系我们”栏目需要特殊显示（含留言板），其余栏目均直接显示置顶的文章。因此，在 ShowArtServlet 中，几类栏目的细节处理略有不同。各主要资源的代码如下。

1. ShowArtServlet 类

程序清单　src\com\servlet\ShowArtServlet.java

```
1：package com.servlet;
2：
3：//import 代码段限于篇幅，略去
4：.....
5：public class ShowArtServlet extends HttpServlet{
6：    public void doGet(HttpServletRequest request,
     HttpServletResponse response) throws ServletException, IOException{
7：        doPost(request,response);
8：   }
9：public void doPost(HttpServletRequest request,
  HttpServletResponse response) throws ServletException, IOException{
10：        String method = request.getParameter("method");
11：        if(method==null||method.equals("")){
12：            response.setContentType("text/html;charset=utf-8");
13：            PrintWriter out = response.getWriter();
14：            HttpSession session = request.getSession();
15：            session.setAttribute("user",null);
16：            session.setAttribute("reSubmit", "no");
17：            out.print("<script>window.parent.location.
                    reload('index.jsp')</script>");
18：            out.flush();
19：            out.close();
20：            return;
21：        }else if(method.equals("listArt")){
22：            //列出新闻速递等标题列表
23：            listArt(request,response);
24：        }else if(method.equals("oneArticle")){
25：            showArticleDetail(request,response);
26：        }
27：  }
28：  private void showArticleDetail(HttpServletRequest request,
```

```
    HttpServletResponse response) throws ServletException, IOException{
29：        response.setContentType("text/html;charset=utf-8");
30：        PrintWriter out = response.getWriter();
31：        int id = Integer.parseInt(request.getParameter("art_id"));
32：        ArticleService as = new ArticleService();
33：        request.setAttribute("Article",as.getArtById(id));
34：        SystemService sy = new SystemService();
35：        request.setAttribute("Sy",sy.getSystemInfoById("1"));
36：        RequestDispatcher rd = request.getRequestDispatcher(
                    "pages/showArticleDetail.jsp");
37：        rd.forward(request,response);
38：          out.flush();
39：           out.close();
40：  }
41：private void listArt(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException{
42：        //列出栏目下的文章列表或直接显示唯一的一篇文章
43：        response.setContentType("text/html;charset=utf-8");
44：        PrintWriter out = response.getWriter();
45：        HttpSession session = request.getSession();
46：        session.setAttribute("reSubmit", "no");
47：        String sort_id = request.getParameter("sort_id");
48：        String group_id=request.getParameter("group_id");
49：        String id=request.getParameter("art_id");   //文章 id
50：        String pages = request.getParameter("pages");
51：        ArticleService as = new ArticleService();
52：        if(pages==null)pages="1";
53：        if (sort_id==null) sort_id="1";
54：        if (group_id==null) group_id="1";
55：        request.setAttribute("listSort",
                  as.listSortNameByGroup(group_id));
56：        String show_url="";
57：        if (group_id.equals("2")&&request.getParameter("note")==null)
58：        {      request.setAttribute("listArt",
              as.listArtInOnePage(group_id,sort_id,
                                  Integer.parseInt(pages), 20));
59：           request.setAttribute("pages",ToolMethod.getPageLinkStr(
          as.getSum("article","art_sort",sort_id),20,
          Integer.parseInt(pages),"ShowArtServlet?
          method=listArt&group_id="+group_id
                  +"&sort_id="+sort_id+"&pages="));
60：        show_url="pages/showArticleList.jsp";
61：        }
62：        else{//除第 2 组外，其余为一级栏目
63：              if (id!=null) //指定显示某篇
64：                    request.setAttribute("article",
```

```
                        as.getArtById(Integer.parseInt(id)));
65：                else //显示栏目下属的第一篇文章
66：                    request.setAttribute("article",
                        as.getFirstArt(group_id,sort_id));
67：            if (group_id.equals("4"))   //“联系我们”特殊页面
68：                show_url="pages/contactUs.jsp";
69：            else{//直接显示，界面与 showArticleDetail.jsp 不同
70：                show_url="pages/showArtDetailDirect.jsp";
71：            }
72：        }
73：        //获取版权信息
74：        SystemService sy = new SystemService();
75：        request.setAttribute("Sy",sy.getSystemInfoById("1"));
76：        RequestDispatcher rd = request.getRequestDispatcher(show_url);
77：          rd.forward(request,response);
78：          out.flush();
79：           out.close();
80：  }
81：}
```

2．ArticleDaoImpl 类的相关代码

```
相关代码　src\com\dao\impl\ArticleDaoImpl.java
1：package com.dao.impl;
2：.....
3：public class ArticleDaoImpl extends BaseDaoImpl implements ArticleDao{
4：   //根据文章 ID 查找文章信息
5：   public Article getArtById(int id)throws SQLException{
6：        PreparedStatement ps = connection.prepareStatement(
                "select * from article where art_id = ?");
7：        ps.setInt(1,id);
8：        ResultSet rs = ps.executeQuery();
9：        Article art = new Article();
10：       if (rs.next()){
11：            art.setArt_id(rs.getInt("art_id"));
12：            art.setArt_title(rs.getString("art_title"));
13：            art.setArt_sort(rs.getString("art_sort"));
14：            art.setArt_sortname(rs.getString("art_sortname"));
15：            art.setArt_IssueTime(rs.getString("art_IssueTime").
                        substring(0,10));
16：            art.setArt_content(rs.getString("art_content"));
17：            art.setArt_group_id(rs.getString("art_group_id"));
18：       }
19：       rs.close();
20：       ps.close();
21：       return art;
```

```
22:  }
23: //获取一级栏目的置顶文章 ， 如公司简介等
24:  public Article getFirstArt(String group_id,String sort_id)
       throws SQLException{
25:            PreparedStatement ps = connection.prepareStatement("
               select * from article where art_top=
                    (select max(art_top) from article where
                    t_sort=? and art_group_id=?)
       and art_sort=? and art_group_id=?");
26:        ps.setString(1, sort_id);
27:        ps.setString(2, group_id);
28:        ps.setString(3, sort_id);
29:        ps.setString(4, group_id);
30:        ResultSet rs = ps.executeQuery();
31:        Article art = new Article();
32:        if(rs.next()){
33:              art.setArt_id(rs.getInt("art_id"));
34:              art.setArt_content(rs.getString("art_content"));
35:              art.setArt_title(rs.getString("art_title"));
36:              art.setArt_IssueTime(rs.getString("art_IssueTime"));
37:              art.setArt_sortname(rs.getString("art_sortname"));
38:              art.setArt_group_id(group_id);
39:              art.setArt_sort(sort_id);
40:        }
41:        rs.close();
42:        ps.close();
43:        return art;
44:  }
45: //获取某个组别的所有栏目名称
46: public List listSortNameByGroup(String group_id)throws SQLException {
47:  String sql = "select * from article where art_group_id=?
                  group by art_sort order by art_sort asc";
48:       PreparedStatement ps = connection.prepareStatement(sql);
49:  ps.setString(1,group_id);
50:  ResultSet rs = ps.executeQuery();
51:  List array = new ArrayList();
52:  while(rs.next()){
53:        Article ar=new Article();
54:        ar.setArt_sortname(rs.getString("art_sortname"));
55:        ar.setArt_group_id(group_id);
56:        ar.setArt_sort(rs.getString("art_sort"));
57:        array.add(ar);
58:  }
59:  rs.close();
```

```
60：  ps.close();
61：  return array;
62：   }
63：//根据组号、栏目号获取某页的所有文章列表
64：    public List listArtPage(String group_id,String sort_id,
                    int curPage, int perpage)throws SQLException{
65：//方法体代码请参考 15.3.1 节的相关内容
66：}
67：//其他方法略
68：......
69：}
```

3．文章列表显示页面 showArticleList.jsp

```
关键代码　WebRoot\pages\showArticle.jsp
1：<!-- 页面左侧的栏目名称列表-->
2：  <ul>
3：       <c:forEach var="sort" items="${listSort}">
4：         <li>
5：         <a class="li_5" href="ShowArtServlet?method=listArt&
       sort_id=${sort.art_sort}&group_id=${sort.art_group_id}">
        ${sort.art_sortname}
6：          </a></li>
7：     </c:forEach>
8：    </ul>
9：  .....
10：<!-- 页面右侧的文章标题列表-->
11：  <ul>
12：    <c:forEach var="art" items="${listArt}">
13：        <li>${art.art_IssueTime}
14：           <a href="ShowArtServlet?method=oneArticle
                &art_id=${art.art_id}" target="_blank">
15：                ${art.art_title}
16：              </a>
17：           </li>
18：         </c:forEach>
19：       </ul>
20：     <div style="float:right; margin-top:20px;">
21：          ${pages}
22：     </div>
```

其他显示页面的内容请直接参考本书所附的源码，不再一一列出。

15.3.3　在线留言模块

前台的在线留言模块仅为用户提供留言信息的输入，相关资源如表 15-3 所示。

表 15-3　前台在线留言模块主要资源

编　　号	类名/文件名	相 关 方 法	功 能 描 述
1	ShowMessageServlet 类	inputMsg：将请求转发到留言板输入页面 addMsg：保存留言信息	留言添加处理
2	MessageService 类 MessageDaoImpl 类 MessageDao 接口	addMsgSave	保存留言信息
3	Message	所有 set/get 方法	封装了留言信息的实体类
4	message_board.jsp		显示留言输入页面
5	RandomServlet 类		生成随机验证码
6	Message.js	CheckMsg(form)：验证表单数据的有效性 checkRND(rnd)：检查验证码正确与否	表单数据的客户端验证

1．ShowMessageServlet 的主要代码

主要代码 src\com\servlet\ShowMessageServlet.java

```
1：package com.servlet;
2：...
3：public class ShowMessageServlet extends HttpServlet {
4：//此处略去部分代码
5：    public void doPost(HttpServletRequest request,
                                    HttpServletResponse response)
6：            throws ServletException, IOException {
7：        String method = request.getParameter("method");
8：        if (method.equalsIgnoreCase("Input")) {
9：            inputMsg(request, response);
10：       }else
11：       if (method.equalsIgnoreCase("Add")) {
12：           addMsg(request, response);
13：       }
14：  }
15：
16：private void addMsg(HttpServletRequest request,
HttpServletResponse response)throws ServletException, IOException {
17：       try {
18：           response.setContentType("text/html;charset=utf-8");
19：           PrintWriter out = response.getWriter();
20：           String msg_company = ToolMethod.encodeSqlHTML(request
21：                   .getParameter("company"));
22：           String msg_industry = ToolMethod.encodeSqlHTML(request
23：                   .getParameter("industry"));
```

```
24:             //读取用户所填写的其他留言信息，代码略
25:             ....
26:                     Message msg = new Message();
27:                     msg.setMsg_company(msg_company);
28:                     msg.setMsg_area(msg_area);
29:                     msg.setMsg_content(msg_content);
30:                     msg.setMsg_fax(msg_fax);
31:                     msg.setMsg_industry(msg_industry);
32:                     msg.setMsg_mail(msg_mail);
33:                     msg.setMsg_name(msg_name);
34:                     msg.setMsg_phone(msg_phone);
35:                     msg.setMsg_role(msg_role);
36:                     msg.setMsg_time(msg_time);
37:                     MessageService ms = new MessageService();
38:                     String resultPage;
39:                     if (ms.addMsgSave(msg) == 1)
40:                             resultPage="pages/msgAddSuccess.jsp";
41:                     else
42:                             resultPage="pages/msgAddFailure.jsp";
43:                     request.setAttribute("ok", "index.jsp");
44:                     RequestDispatcher rd = request
                                .getRequestDispatcher(resultPage);
45:                     rd.forward(request, response);
46:                     out.flush();
47:                     out.close();
48:                 }
49:         } catch (Exception e) {
50:                 e.printStackTrace();
51:                 response.setContentType("text/html;charset=utf-8");
52:                 PrintWriter out = response.getWriter();
53:                 out.print("<script>window.parent.location.
                                reload('index.jsp')</script>");
54:                 out.flush();
55:                 out.close();
56:         }
57: }
58:
59: public void inputMsg(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException{
60:         response.setContentType("text/html;charset=utf-8");
61:         PrintWriter out = response.getWriter();
62:         // 版权信息
63:         SystemService sy = new SystemService();
64:         request.setAttribute("Sy", sy.getSystemInfoById("1"));
65:         RequestDispatcher rd = request
66:                         .getRequestDispatcher("pages/message_board.jsp");
```

```
67：          rd.forward(request, response);
68：          out.flush();
69：          out.close();
70：  }
71：}
```

2．MessageDaoImpl 类的相关代码

```
相关代码　src\com\dao\impl\MessageDaoImpl.java
1：package com.dao.impl;
2：…
3：public class MessageDaoImpl extends BaseDaoImpl implements MessageDao{
4：public boolean addMsgSave(Message message)throws SQLException{
5：   String sql = "insert into message(msg_company,msg_industry,
                    msg_area,msg_time,msg_name,msg_phone,
                    msg_fax,msg_mail,msg_content,issueTime)
                    values (?,?,?,?,?,?,?,?,?,now())";
6：PreparedStatement ps = connection.prepareStatement(sql);
7：    ps.setString(1,message.getMsg_company());
8：    ps.setString(2,message.getMsg_industry());
9：    ps.setString(3,message.getMsg_area());
10：  ps.setString(4,message.getMsg_time());
11：  ps.setString(5,message.getMsg_name());
12：  ps.setString(6,message.getMsg_phone());
13：  ps.setString(7,message.getMsg_fax());
14：  ps.setString(8,message.getMsg_mail());
15：  ps.setString(9,message.getMsg_content());
16：  ps.executeUpdate();
17：  ps.close();
18：  return true;
19：  }
20：}
```

3．message_board.jsp 的主要代码

```
关键代码　WebRoot\pages\message_board.jsp
1：<script src="pages/js/Message.js" type="text/JavaScript"></script>
2：  ...
3：<form action="ShowMessageServlet?method=Add"
       onsubmit="return CheckMsg(this);" method="post">
4：      <ul>
5：        <li>公司名称：
6：          <input name="company" type="text" size="15"/>
7：        </li>
8：         .....<!--其他输入项-->
9：        验证码：
10：        <input name="rnd" id="rnd" type="text" size="13" />
11：       <img src="imgcode" id="rc" onclick="this.src=this.src+'?chg='+
```

```
            new Date()" title="看不清？单击换一张图片" />
12：    </ul>
13：</form>
```

理解该页面的 3 个关键点。

- 调用了 JS 函数完成表单的客户端数据验证，其中随机码的验证采用了 Ajax 技术。
- 调用了 RandomServlet 生成随机验证码，访问路径是/imgcode，单击图片时通过生成一个不重复的图片地址迫使其重新从服务器取得新的验证码。
- 表单数据提交给 ShowMessageServlet 处理，由其中的 addMsgSave 方法完成留言信息的添加。

15.4　后台功能模块

后台的所有功能只对管理员开放，因此需登录后才能使用。

15.4.1　登录模块

后台的所有资源均使用“/admin”的访问路径进行访问，其中所有 Servlet 的访问路径均以“/admin”开头，而页面文件则直接存储于 WebRoot\admin 目录下。因此，为了对这些资源的访问权限进行控制，可为后台资源设置通用的过滤器，通过过滤器检查用户是否已经登录。项目的配置文件 web.xml 中相关的配置如下：

```
相关配置段　WebRoot\WEB-INF\web.xml
1：<filter>
2：        <filter-name>loginFilter</filter-name>
3：        <filter-class>com.util.loginFilter</filter-class>
4：    </filter>
5：    <filter-mapping>
6：        <filter-name>loginFilter</filter-name>
7：        <url-pattern>/admin/*</url-pattern>
8：    </filter-mapping>
```

登录模块主要的相关资源如表 15-4 所示。

表 15-4　后台登录模块主要资源

编　　号	类名/文件名	相 关 方 法	功 能 描 述
1	LoginFilter 过滤器类	doFilter	对/admin 下的所有访问请求进行过滤，当用户未登录时，将请求重定向至登录页面 login.jsp
2	login.jsp		显示登录界面，数据提交至 LoginServlet 类进行验证处理
3	LoginServlet 类	doPost	调用 AdminService 类的相关方法验证登录信息

续表

编　　号	类名/文件名	相 关 方 法	功 能 描 述
4	AdminService 类	login	调用 AdminDao 的 getAdminByField 方法，检测用户名与密码是否正确
5	AdminDaoImpl 类 AdminDao 接口	getAdminByField	根据条件读取符合条件的管理员用户信息
6	Admin	所有 set/get 方法	封装了管理员用户信息的实体类

1．LoginFilter 过滤器类

主要代码　src\com\util\LoginFilter.java

```
1：package com.util;
2：.....
3：public class LoginFilter implements Filter{
4：        FilterConfig config = null;
5：        public void init(FilterConfig config) throws ServletException {
6：              this.config = config;
7：        }
8：
9：        public void doFilter(ServletRequest request, ServletResponse response,
10：                                        FilterChain chain)
11：  throws IOException, ServletException {
12：           response.setContentType("text/html;charset=utf-8");
13：              HttpSession session = ((HttpServletRequest)request).getSession();
14：              Admin user = (Admin)session.getAttribute("user");
15：              if(user==null||user.equals("")){
16：                 HttpServletResponse res = (HttpServletResponse) response;
17：                 HttpServletRequest req = (HttpServletRequest) request;
18：                 res.sendRedirect(req.getContextPath()+"/login.jsp");
19：              }else{
20：                 chain.doFilter(request, response);
21：              }
22：        }
23：        public void destroy() {
24：              this.config = null;
25：        }
26：}
```

2．login.jsp 登录页面

主要代码 WebRoot\login.jsp

```
1：<form method="post" onsubmit="return checkLogin(this);" action="login">
2：          <input type="text" name="name" />
3：          <input type="password" name="pass" />
4：          <span>${info}</span>
5：        ...
```

```
6：  </form>
```

此处删去了 CSS 代码，完整的页面源码请参考本书相关资料。这里的表单数据将提交给 URL 为 login 的 Servlet 处理，即 com.servlet.LoginServlet 类。EL 表达式“${info}”表示来自验证类 LoginServlet 的错误提示信息。

3．LoginServlet 类

```
主要代码　src\com\servlet\loginServlet.java
7：package com.servlet;
8：.....
9：public class LoginServlet extends HttpServlet{
10：
11：public void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
12：        doPost(request, response);
13：  }
14：
15：public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
16：  response.setContentType("text/html;charset=utf-8");
17：  PrintWriter out = response.getWriter();
18：  String info ="";   //返回登录页面的出错信息
19：      //下面略去了验证数据有效性的代码
20：      .....
21：  String admin_name2 = ToolMethod.encodeLoginStr(
                        request.getParameter("name"));
22：  String admin_pw2 = ToolMethod.encodeLoginStr(
                        request.getParameter("pass"));
23：  Admin admin = new Admin();
24：  admin.setAdmin_name(admin_name2);
25：  admin.setAdmin_pwd(MDFive.getMD5(admin_pw2.getBytes()));
26：  AdminService us = new AdminService();
27：  int b = us.login(admin);
28：  switch(b){
29：  case 1:
30：       HttpSession session = request.getSession();
31：       session.setAttribute("user", us.getAdminByField(admin_name));
32：      response.sendRedirect("admin/admin.jsp");
33：      return;
34：       case 2:
35：            info ="账号不存在";
36：            break;
37：       case 3:
38：            info ="密码不正确";
39：            break;
40：       }
```

```
41：      request.setAttribute("info", info);
42：      RequestDispatcher rd = request.getRequestDispatcher("login.jsp");
43：  rd.forward(request, response);
44：  out.flush();
45：  out.close();
46：    }
47：}
```

该 Servlet 调用了 AdminService 服务类的 login 方法，该方法通过调用 AdminDaoImpl 中的 getAdminByField 方法查询数据库，返回值为整数，表示所得到的不同的检验结果。

4．AdminService 类的相关代码

```
相关代码  src\com\service\AdminService.java
48：package com.service;
49：......
50：public class AdminService {
51：private AdminDao admin_dao = new AdminDaoImpl();
52：......
53：public int login(Admin admin){
54：  Connection connection = null;
55：  try {
56：        connection = Database.getConnection();
57：        admin_dao.setConnection(connection);
58：        Admin userDB = admin_dao.getAdminByField("Admin_name",
                  admin.getAdmin_name());
59：              if(userDB == null){
60：                    return 2;
61：              }else
62：                if(userDB.getAdmin_pwd().equals(admin.getAdmin_pwd())){
63：                    return 1;
64：              }else{
65：                    return 3;
66：              }
67：        } catch (Exception e) {
68：              e.printStackTrace();
69：              Database.rollback();
70：              return 0;
71：        } finally {
72：              Database.releaseConnection(connection);
73：        }
74：        }
75：}
```

5．AdminDaoImpl 类的相关代码

```
相关代码   src\com\dao\impl\AdminDaoImpl.java
1：package com.dao.impl;
2：......
3：public class AdminDaoImpl extends BaseDaoImpl implements AdminDao{
```

```
4:  public Admin getAdminByField(String fieldName, String fieldValue)
                                        throws SQLException {
5:      PreparedStatement ps = connection.prepareStatement(
            "SELECT * FROM t_admin where " + fieldName + " = ?");
6:      ps.setString(1, fieldValue);
7:      ResultSet rs = ps.executeQuery();
8:      Admin admin = new Admin();
9:      while (rs.next()) {
10:         admin.setAdmin_id(rs.getInt("admin_id"));
11:         admin.setAdmin_name(rs.getString("admin_name"));
12:         admin.setAdmin_pwd(rs.getString("admin_pwd"));
13:         admin.setAdmin_role(rs.getInt("admin_role"));
14:         return admin;
15:     }
16:     rs.close();
17:     ps.close();
18:     return null;
19: }
20: .....
21: }
```

在 AdminService 类的 login 方法中调用 AdminDaoImpl 类的 getAdminByField 方法时，第 1 个参数被设为 Admin_name，表示根据 Admin_name 列的值来查找符合条件的记录。

15.4.2　文章管理模块

管理员可在后台完成对各栏目文章的增、删、查、改操作。这里的“文章”概念是广义的，包括“公司信息”组、“业务信息”组及“文章信息”组下的所有栏目的文章信息。该模块的相关资源如表 15-5 所示。

表 15-5　后台文章管理模块主要资源

编　　号	类名/文件名	相 关 方 法	功 能 描 述
1	ArtServlet 类	doPost	根据参数 method 分发处理请求
		listArt	获取显示文章列表所需的数据，并将请求转发至后台的文章列表显示页面 article_index.jsp
		showArtAddPage	预处理并将请求转发到添加文章页面 article_add.jsp
		artAddSave	保存添加的文章信息
		showArtEditPage	预处理并将请求转发到文章修改页面 article_edit.jsp
		artEditSave	保存修改后的文章信息
		delArt	删除文章(一篇或多篇)
		doTop	文章置顶处理

续表

编　　号	类名/文件名	相 关 方 法	功 能 描 述
2	ArticleService ArticleDaoImpl ArticleDao	getFirstArt	获取某个栏目下的置顶文章信息
		getArtById	根据文章 ID 获取文章信息
		listArtPage	显示某页的文章列表
		addArtSave	保存新增的文章信息
		editArtSave	保存修改后的文章信息
		listSortNameByGroup	根据组号获取所有的栏目名称
3	Article	所有 set/get 方法	封装了文章信息的实体类
4	article_index.jsp	文章管理首页	显示文章标题列表及增、删、查、改链接
5	article_add.jsp	添加文章页面	添加新文章的输入页面
6	article_edit.jsp	修改文章页面	可修改文章的标题、内容等信息

当用户登录成功后，将在后台首页显示各栏目的导航链接，通过单击栏目名称，将请求发送到 ArtServlet，该 Servlet 调用其 listArt 方法处理请求，并将请求转发至显示文章列表页面 article_index.jsp；该页面上提供了文章的增、删、查、改链接，通过单击不同的链接，再次将请求发送到 ArtServlet 中，其 doPost 方法将根据参数 method 的不同值调用不同的方法完成请求处理。下面列出本模块中主要程序文件的相关代码。

1．ArtServlet 类

主要代码　src\com\servlet\ArtServlet.java

```
1：package com.servlet;
2：.....
3：public class ArtServlet extends HttpServlet {
4：public void doPost(HttpServletRequest request,
   HttpServletResponse response) throws ServletException, IOException{
5：        String method = request.getParameter("method");
6：         //此处略去关于参数 method 有效性检测的部分代码
7：           .....
8：        if (method.equals("addArtSave")) {
9：              addArtSave(request, response);
10：       } else if (method.equals("showArtAddPage")) {
11：             showArtAddPage(request, response);
12：       } else if (method.equals("showArtEditPage")) {
13：             showArtEditPage(request, response);
14：       } else if (method.equals("editArtSave")) {
15：             editArtSave(request, response);
16：       } else if (method.equals("delArt")) {
17：             delArt(request, response);
18：       } else if (method.equals("listArt")) {
19：             listArt(request, response);
20：       } else if (method.equals("doTop")) {
21：             doTop(request, response);
22：       } else {
```

```
23：            //转发至登录页面 login.jsp，代码略
24：                    ......
25：          }
26：  }
27：
28：// 保存新文章
29：private void addArtSave(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException{
30：  response.setContentType("text/html;charset=utf-8");
31：  PrintWriter out = response.getWriter();
32：  String sort_id = request.getParameter("art_sort");
33：  String group_id = request.getParameter("group_id");
34：  String backPath = "ArtServlet?method=listArt&group_id=" + group_id
35：                    + "&sort_id=" + sort_id;
36：  // 略去重复提交问题的解决
37：          ......
38：        ToolMethod tool = new ToolMethod();
39：        String art_title = ToolMethod.encodeHTML(
    tool.convertStringNull(request.getParameter("art_title")));
40：  String content = request.getParameter("art_content").trim();
41：  if (art_title == null || art_title.equals("") || content == null
42：  || content.equals("") || sort_id == null || sort_id.equals("")){
43：  request.setAttribute("sorry", backPath);
44：  RequestDispatcher rd = request.getRequestDispatcher("sorry.jsp");
45：  rd.forward(request, response);
46：  return;
47：} else {
48：  ArticleService as = new ArticleService();
49：     //获取与栏目编号对应的栏目名称
50：       String art_sortname =as.getFirstArt(group_id,
                                        sort_id).getArt_sortname();
51：  Article art = new Article();
52：  art.setArt_title(art_title);
53：  art.setArt_content(content);
54：  art.setArt_sort(sort_id);
55：  art.setArt_sortname(art_sortname);
56：  art.setArt_group_id(group_id);
57：  if (as.addArtSave(art)) {
58：          request.setAttribute("ok", backPath);
59：          RequestDispatcher rd = request
60：                  .getRequestDispatcher("success.jsp");
61：          rd.forward(request, response);
62：  } else {
63：          request.setAttribute("sorry", backPath);
64：          RequestDispatcher rd = request
65：                  .getRequestDispatcher("sorry.jsp");
```

```
66：            rd.forward(request, response);
67：  }
68：  }
69：  out.flush();
70：  out.close();
71：}
72：//显示文章添加页面前的处理
73：private void showArtAddPage(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
74：    PrintWriter out = response.getWriter();
75：    response.setContentType("text/html;charset=utf-8");
76：    HttpSession session = request.getSession();
77：    session.setAttribute("reSubmit", "no");
78：    String group_id = request.getParameter("group_id");
79：    String sort_id = request.getParameter("sort_id");
80：    String method = "listArt&group_id=" + group_id;
81：    String back_url = "ArtServlet?method=" + method;
82：    String url = "../admin/ArtServlet?method=addArtSave";
83：    ArticleService as = new ArticleService();
84：    request.setAttribute("ListSelect", as.listSortNameByGroup(group_id));
85：    request.setAttribute("group_id", group_id);
86：    request.setAttribute("sort_id", sort_id);
87：    request.setAttribute("save_url", url);
88：    request.setAttribute("back_url", back_url);
89：    request.setAttribute("method", method);
90：    RequestDispatcher rd = request.getRequestDispatcher("article_add.jsp");
91：    rd.forward(request, response);
92：    out.flush();
93：    out.close();
94：}
95：//文章修改页面的显示及保存处理与添加新文章相似，限于篇幅，此处略去
96：.....
97：private void delArt(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
98：  response.setContentType("text/html;charset=utf-8");
99：  PrintWriter out = response.getWriter();
100：        String id[] = request.getParameterValues("id");
101：        String group_id = request.getParameter("group_id");
102：        String sort_id = request.getParameter("sort_id");
103：        ArticleService as = new ArticleService();
104：        // 删除文章中涉及的上传图片
105：        for (int i = 0; i < id.length; i++) {
106：            Article ar = as.getArtById(Integer.parseInt(id[i]));
107：            String artcontent = ar.getArt_content();
108：            Pattern pattern = Pattern.compile("/upload/File/(.+?)\"",
109：                    Pattern.CASE_INSENSITIVE);
```

```
110：              Matcher matcher = pattern.matcher(artcontent);
111：             while (matcher.find()) {
112：                  String path = matcher.group();
113：                  String fileName = path.substring(path.lastIndexOf("/") + 1,
114：                            path.indexOf("\""));
115：                   String realpath = this.getServletContext().getRealPath("/")
116：                              + "upload\\File\\";
117：                    File file = new File(realpath, fileName);
118：                    if (file.exists()) {
119：                          file.delete();
120：                    }
121：               }
122：         }
123：         // ------删除文章-------
124：         BaseService bs = new BaseService();
125：         bs.del(id, "article", "art_id");
126：         String method = "listArt&group_id=" + group_id + "&sort_id=" + sort_id;
127：         String backPath = "ArtServlet?method=" + method;
128：         RequestDispatcher rd = request.getRequestDispatcher(backPath);
129：         rd.forward(request, response);
130：         out.flush();
131：         out.close();
132：         }
133：
134：  //显示栏目文章列表
135：  private void listArt(HttpServletRequest request,
       HttpServletResponse response) throws ServletException, IOException {
136：         response.setContentType("text/html;charset=utf-8");
137：         PrintWriter out = response.getWriter();
138：         HttpSession session = request.getSession();
139：         session.setAttribute("reSubmit", "no");
140：         String sort_id = request.getParameter("sort_id");
141：         String group_id = request.getParameter("group_id");
142：         ArticleService as = new ArticleService();
143：         String sortname = as.getFirstArt(group_id, sort_id).
                                                  getArt_sortname();
144：         String method = "listArt";
145：         String pages = request.getParameter("pages"); //  页号
146：         String params="&group_id="+ group_id + "&sort_id=" + sort_id;
147：         String add_url = "ArtServlet?method=showArtAddPage"+params;
148：         String edit_url = "ArtServlet?method=showArtEditPage"+params;
149：         String del_url = "ArtServlet?method=delArt"+params;
150：         String show_url = "../ShowArtServlet?note=1&method=listArt"+params;
151：         if (pages == null)
152：           pages = "1";
153：           request.setAttribute("ListArt", as.listArtPage(
```

```
                    group_id, sort_id, Integer.parseInt(pages), 8));
154：          request.setAttribute("pages",ToolMethod.getPageLinkStr(
          as.getSum("article","art_sort", sort_id), 8,
          Integer.parseInt(pages),"ArtServlet?method=listArt&pages="));
155：          request.setAttribute("sort_id", sort_id);
156：        request.setAttribute("sortname", sortname);
157：        request.setAttribute("add_url", add_url);
158：        request.setAttribute("edit_url", edit_url);
159：        request.setAttribute("del_url", del_url);
160：        request.setAttribute("params", params);
161：        request.setAttribute("show_url", show_url);
162：        request.setAttribute("group_id", group_id);
163：        RequestDispatcher rd = request.getRequestDispatcher
                            ("article_index.jsp");
164：        rd.forward(request, response);
165：        out.flush();
166：        out.close();
167：   }
168：
169：   //文章置顶处理
170：private void doTop(HttpServletRequest request,
  HttpServletResponse response) throws ServletException, IOException {
171：            response.setContentType("text/html;charset=utf-8");
172：            PrintWriter out = response.getWriter();
173：            String id[] = request.getParameterValues("id");
174：            String params=request.getParameter("params");
175：            ArticleService as = new ArticleService();
176：            as.doTop(id);
177：            RequestDispatcher rd = request.getRequestDispatcher(
                        "ArtServlet?method=listArt"+params );
178：            rd.forward(request, response);
179：            out.flush();
180：            out.close();
181：        }
182：}
```

2．ArticleDaoImpl 类

该类的几个查询方法已经在前面的章节中介绍过，这里仅列出保存新文章和保存修改后的文章的两个方法的代码，如下所示：

```
相关代码   src\com\dao\impl\ArticleDaoImpl.java
1：package com.dao.impl;
2：.....
3：public class ArticleDaoImpl extends BaseDaoImpl implements ArticleDao{
4：
5：//保存添加的新文章信息
6：  public boolean addArtSave(Article art) throws SQLException {
```

```
7：   PreparedStatement ps = connection.prepareStatement(
          "insert into article(art_title, art_sort, art_sortname,
          art_content,art_group_id,art_IssueTime,art_top)
          values(?,?,?,?,?,now(),now())");
8：   ps.setString(1,art.getArt_title());
9：   ps.setString(2,art.getArt_sort());
10：  ps.setString(3,art.getArt_sortname());
11：  ps.setString(4,art.getArt_content());
12：  ps.setString(5,art.getArt_group_id());
13：  ps.executeUpdate();
14：  ps.close();
15：  return true;
16：   }
17：
18：//保存修改后的文章信息
19：public boolean editArtSave(String id,Article art)throws SQLException{
20：  PreparedStatement ps = connection.prepareStatement(
          "update article set art_title=?,art_sort=?,art_sortname=?,
           art_content=?,art_IssueTime=now() where art_id=?");
21：  ps.setString(1,art.getArt_title());
22：  ps.setString(2,art.getArt_sort());
23：  ps.setString(3,art.getArt_sortname());
24：  ps.setString(4,art.getArt_content());
25：  ps.setString(5,id);
26：  ps.executeUpdate();
27：  ps.close();
28：  return true;
29：   }
30：}
```

3．文章管理页面 article_index.jsp（数据展示部分）

```
关键代码　WebRoot\admin\article_index.jsp
1：<%@ page language="java" isErrorPage="true" pageEncoding="UTF-8"%>
2：<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
3：.....
4：    <div class="head">${sortname}</div>
5：     <div class="head2">
6：       <ul>
7：         <li><a href="${add_url}" target="right">添加文章</a></li>
8：       </ul>
9：     </div>
10：    <!--content-->
11：   <form name="dotop" action="${del_url}" name="artList"
                onsubmit="return checkCheck();" method="post">
12：   <div id="right_content">
13：          <div class="part_top">
14：             <ul>
15：                <li class="li_1"><a>ID</a></li>
```

```
16：                <li class="li_2"><a>选择</a></li>
17：                <li class="li_3"><a>文章标题</a></li>
18：                <li class="li_4"><a>发布时间</a></li>
19：                <li class="li_5"><a>设置</a></li>
20：              </ul>
21：            </div>
22：            <c:forEach var="art" items="${ListArt}">
23：            <div class="part_content">
24：              <ul>
25：                <li><a>${art.art_id}</a></li>
26：                <li><a>
27：                  <input type="checkbox" name="id" value="${art.art_id}"/>
28：                  </a></li>
29：                <li><a>${art.art_title}</a></li>
30：                <li><a>${art.art_IssueTime}</a></li>
31：                <li>
32：                <a href="${show_url}&art_id=${art.art_id}"
                              target="_blank">查看</a>
33：                <a href="${edit_url}&art_id=${art.art_id}" target="right">
34：                    修改</a>
35：                <a href="#" onclick="delLine('${del_url}&id=${art.art_id}')">
36：                  删除</a></li>
37：              </ul>
38：            </div>
39：       </c:forEach>
40：            <div class="part_bt">
41：              <ul>
42：                <li class="li_6"><a href="#" onclick="selectAll();">
43：                [全选] </a>
44：                  <a href="#" onclick="turnOver();"> [反选] </a></li>
45：                <li class="li_7"><input type="submit" value="删除选中"/>
46：                <input type="button" onclick="doTop('${params}')"
47：                        value="置顶"/></li>
48：                <li> ${pages}</li>
49：              </ul>
50：    </div>
51：</form>
```

4．文章修改页面 article_edit.jsp（数据展示部分）

```
关键代码　WebRoot\admin\article_edit.jsp
1：<form action="${save_url}" onsubmit="return checkArt(this);"
        method="post">
2：<div class="part_content" id="part_content_a">
3：      <ul>
4：        <li><a>内容标题:</a></li>
5：        <li><a>
6：        <input name="art_title" value="${UpdateArt.art_title }" size="40"/>
```

```
7：          </a></li>
8：        </ul>
9：   </div>
10：   <div class="part_content">
11：      <ul>
12：         <li><a>所属栏目:</a></li>
13：         <li id="readonly_sort_name">${UpdateArt.art_sortname}</li>
14：      </ul>
15：    </div>
16：<div class="part_content" id="part_content_a">
17：    <ul>
18：       <li><a>修改内容:</a></li>
19：       <li><a>
20：                <FCK:editor id="art_content" basePath="../FCKeditor/"
21：                          defaultLanguage="zh-cn"
22：                          tabSpaces="8" ...>   //略去部分属性列表
23：                        ${UpdateArt.art_content}
24：                </FCK:editor>
25：              </a></li>
26：   <input type="hidden" name="art_id" value="${UpdateArt.art_id }">
27：   <input type="hidden" name="art_sort" value="${UpdateArt.art_sort}">
28：   <input type="hidden" name="group_id"
                                  value="${UpdateArt.art_group_id}">
29：      </ul>
30：    </div>
31：  ......
32：</form>
33：......
```

文章添加页面 article_add.jsp 的代码与修改页面 article_edit.jsp 的代码相似，此处不再列出，请参考本书所附资源。

15.4.3　留言管理模块

管理员对留言信息进行的管理主要包括查看留言信息、修改留言状态为“已阅”、删除留言等几项操作。相关的资源如表 15-6 所示。

表 15-6　留言管理模块主要资源

编　号	类名/文件名	相 关 方 法	功 能 描 述
1	MessageServlet 类	doPost listAllMsg listNewMsg showMsgDetail delMsg editMsg	根据请求的参数类型，调用相关的 Service 类完成留言的显示、状态修改、删除操作

续表

编　号	类名/文件名	相 关 方 法	功 能 描 述
2	MessageService 类	listAll：显示所有留言 listNew：显示待阅留言 getMsgById：根据 ID 读取留言信息 editMsgSave：留言修改保存	调用 MessageDaoImpl 完成显示及删改操作
3	BaseService	del	根据表名及字段值删除某条记录，调用 BaseDaoImpl 实现
4	MessageDaoImpl 类 MessageDao 接口	ListMsg：根据条件列出留言信息列表 GetMsgById：根据 ID 读取留言 editMsgSave：修改留言信息	完成对留言表的查询、修改等具体操作
5	Message		封装了留言信息的实体类
6	allmessages.jsp		显示所有留言列表
7	message_board.jsp		显示所有待阅留言，可设置留言状态为“已阅”
8	message_detail.jsp		显示某条留言的所有信息

该模块的主要代码如下。

1．MessageServlet 类

主要代码　src\com\servlet\MessageServlet.java

```
1：package com.servlet;
2：
3：public class MessageServlet extends HttpServlet {
4：// doGet 方法，直接调用 doPost 方法，代码略
5：......
6：public void doPost(HttpServletRequest request,
 HttpServletResponse response) throws ServletException, IOException {
7：    String method = request.getParameter("method");
8：    String methods = ToolMethod.encodeSqlHTML(method);
9：    boolean sqlDanger = ToolMethod.checkSqlInject(method);
10：  int especial = methods.indexOf("#");
11：  if (method == null || method.equals("") || sqlDanger ||
                especial == 0|| especial == 1) {
12：              // 返回登录页面，代码略去
13：                  ......
14：              return;
15：  } else if (method.equalsIgnoreCase("ListAll")) {
16：        listAllMsg(request, response);
17：  } else if (method.equalsIgnoreCase("ListNew")) {
18：        listNewMsg(request, response);
19：  } else if (method.equalsIgnoreCase("ShowOne")) {
20：        showMsgDetail(request, response);
21：  } else if (method.equalsIgnoreCase("Del")) {
```

```
22：          delMsg(request, response);
23：  } else if (method.equalsIgnoreCase("Edit")) {
24：          editMsg(request, response);
25：  }
26：}
27：
28：private void listAllMsg(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
29：  response.setContentType("text/html;charset=utf-8");
30：  PrintWriter out = response.getWriter();
31：  HttpSession session = request.getSession();
32：  session.setAttribute("reSubmit", "no");
33：  String pages = request.getParameter("pages");
34：  if (pages == null)pages = "1";
35：  MessageService ms = new MessageService();
36：  request.setAttribute("ListAll", ms
37：                                  .listAll(Integer.parseInt(pages), 10));
38：  request.setAttribute("pages", ToolMethod.getPageLinkStr(
                ms.getSum("message"), 10, Integer.parseInt(pages),
39：              "MessageServlet?method=ListAll&pages="));
40：  RequestDispatcher rd = request.getRequestDispatcher(
                                                          "allmessages.jsp");
41：          rd.forward(request, response);
42：          out.flush();
43：          out.close();
44：      }
45：
46：private void listNewMsg(HttpServletRequest request,
47：                  HttpServletResponse response) throws ServletException,
              IOException {
48：  response.setContentType("text/html;charset=utf-8");
49：  PrintWriter out = response.getWriter();
50：  HttpSession session = request.getSession();
51：  session.setAttribute("reSubmit", "no");
52：  String pages = request.getParameter("pages");
53：  if (pages == null) pages = "1";
54：  MessageService ms = new MessageService();
55：  request.setAttribute("ListNew", ms
56：                                  .listNew(Integer.parseInt(pages), 10));
57：  request.setAttribute("pages", ToolMethod.getPageLinkStr(
                        ms.getSum("message", "msg_role", "0"), 10,
                        Integer.parseInt(pages),
58：                       "MessageServlet?method=ListNew&pages="));
59：  RequestDispatcher rd = request
60：                  .getRequestDispatcher("message_board.jsp");
61：  rd.forward(request, response);
```

```
62:  out.flush();
63:  out.close();
64: }
65:
66: private void showMsgDetail(HttpServletRequest request,
67:                  HttpServletResponse response) throws ServletException,
                 IOException {
68:      String ids = ToolMethod.encodeSqlHTML(request.getParameter("id"));
69:  int idd = ids.indexOf("#");
70:  if (ids == null || ids.equals("") || idd == 0 || idd == 1) {
71:      //返回登录页面 login.jsp,代码略
72:        .....
73:  } else if (ids != null && !ids.equals("")) {
74:  response.setContentType("text/html;charset=utf-8");
75:  PrintWriter out = response.getWriter();
76:  HttpSession session = request.getSession();
77:  session.setAttribute("reSubmit", "no");
78:  int id = Integer.parseInt(ids);
79:  MessageService ms = new MessageService();
80:  request.setAttribute("msg", ms.getMsgById(id));
81:  RequestDispatcher rd = request
82:                  .getRequestDispatcher("message_detail.jsp");
83:  rd.forward(request, response);
84:  out.flush();
85:  out.close();
86:  }
87: }
88:
89: private void delMsg(HttpServletRequest request,
90:                      HttpServletResponse response)
91:                  throws ServletException, IOException {
92: try {
93:  response.setContentType("text/html;charset=utf-8");
94:  PrintWriter out = response.getWriter();
95:  HttpSession session = request.getSession();
96:  session.setAttribute("reSubmit", "no");
97:  String id[] = request.getParameterValues("id");
98:  BaseService baseService = new BaseService();
99:  baseService.del(id, "message", "msg_id");
100:        RequestDispatcher rd = request.getRequestDispatcher(
                                  "MessageServlet?method=ListAll");
101:        rd.forward(request, response);
102:        out.flush();
103:        out.close();
104:        } catch (Exception e) {
105:        PrintWriter out = response.getWriter();
```

```
106:        e.printStackTrace(System.err);
107:        out.flush();
108:        out.close();
109:        }
110:    }
111:
112:    private void editMsg(HttpServletRequest request,
113:                         HttpServletResponse response) throws ServletException,
114:                        IOException {
115:        response.setContentType("text/html;charset=utf-8");
116:        PrintWriter out = response.getWriter();
117:        String id = request.getParameter("id");
118:        MessageService ms = new MessageService();
119:        ms.editMsgSave(id);
120:        RequestDispatcher rd = request.getRequestDispatcher(
                                    "MessageServlet?method=ListAll");
121:        rd.forward(request, response);
122:        out.flush();
123:        out.close();
124:        }
125: }
```

2．MessageService 类的相关代码

仅列出 listAll 和 listNew 方法，其他方法是对 MessgeDaoImpl 类中同名方法的封装，不再列出。

相关代码　src\com\service\MessageService.java

```
1: package com.service;
2: ......
3: public class MessageService extends BaseService{
4:    private MessageDao md = new MessageDaoImpl();
5: //   得到未审核的留言
6:    public List listNew(int curPage, int perpage){
7:          Connection connection = null;
8:          try{
9:                connection = Database.getConnection();
10:               md.setConnection(connection);
11:               return md.listMsg(curPage, perpage,"msg_role='0'");
12:         }catch(Exception e){
13:               e.printStackTrace();
14:               Database.rollback();
15:         }finally{
16:               Database.releaseConnection(connection);
17:         }
18:         return new ArrayList();
19:   }
20:   //列出全部留言
```

```
21：  public List listAll(int curPage, int perpage){
22：        Connection connection = null;
23：        try{
24：              connection = Database.getConnection();
25：              md.setConnection(connection);
26：              return md.listMsg(curPage, perpage,null);
27：        }catch(Exception e){
28：              e.printStackTrace();
29：              Database.rollback();
30：
31：        }finally{
32：              Database.releaseConnection(connection);
33：        }
34：        return new ArrayList();
35：  }
36：}
```

3．MessageDaoImpl 类的相关代码

```
相关代码  src\com\dao\impl\MessageDaoImpl.java
1：package com.dao.impl;
2：......
3：public class MessageDaoImpl extends BaseDaoImpl implements MessageDao{
4：   public boolean addMsgSave(Message message)throws SQLException{
5：      .......  //保存新留言信息，代码参考 5.3.3 小节
6：   }
7：
8：   public List listMsg(int curPage, int perpage,String condition)
                  throws SQLException{
9：   List array = new ArrayList();
10：  String sql = "select * from message ";
11：  if (condition!=null)
12：        sql+=" where "+condition;
13：  sql+=" order by msg_id desc limit ?,?";
14：  PreparedStatement ps = connection.prepareStatement(sql);
15：  ps.setInt(1, (curPage-1)*perpage);
16：  ps.setInt(2, perpage);
17：  ResultSet rs = ps.executeQuery();
18：  while(rs.next()){
19：        Message mg = new Message();
20：        mg.setMsg_id(rs.getInt("msg_id"));
21：        mg.setMsg_company(rs.getString("msg_company"));
22：        mg.setMsg_industry(rs.getString("msg_industry"));
23：        mg.setMsg_area(rs.getString("msg_area"));
24：        mg.setMsg_time(rs.getString("msg_time"));
25：        mg.setMsg_name(rs.getString("msg_name"));
26：        mg.setMsg_phone(rs.getString("msg_phone"));
```

```
27：            mg.setMsg_fax(rs.getString("msg_fax"));
28：            mg.setMsg_mail(rs.getString("msg_mail"));
29：            mg.setMsg_content(rs.getString("msg_content"));
30：            mg.setMsg_role(rs.getString("msg_role"));
31：            mg.setIssueTime(rs.getString("issueTime").substring(0,19));
32：            array.add(mg);
33：  }
34：          rs.close();
35：  ps.close();
36：  return array;
37：}
38：
39：public Message getMsgById(int id)throws SQLException{
40：  String sql = "select * from message where msg_id=?";
41：  PreparedStatement ps = connection.prepareStatement(sql);
42：  ps.setInt(1, id);
43：  ResultSet rs = ps.executeQuery();
44：  Message mg = new Message();
45：  if(rs.next()){
46：            mg.setMsg_id(rs.getInt("msg_id"));
47：            mg.setMsg_company(rs.getString("msg_company"));
48：            mg.setMsg_industry(rs.getString("msg_industry"));
49：            mg.setMsg_area(rs.getString("msg_area"));
50：            mg.setMsg_time(rs.getString("msg_time"));
51：            mg.setMsg_name(rs.getString("msg_name"));
52：            mg.setMsg_phone(rs.getString("msg_phone"));
53：            mg.setMsg_fax(rs.getString("msg_fax"));
54：            mg.setMsg_mail(rs.getString("msg_mail"));
55：            mg.setMsg_content(rs.getString("msg_content"));
56：            mg.setMsg_role(rs.getString("msg_role"));
57：            mg.setIssueTime(rs.getString("issueTime").substring(0,19));
58：  }
59：  rs.close();
60：  ps.close();
61：            return mg;
62：   }
63：
64：public boolean editMsgSave(String id)throws SQLException{
65：  String sql = "update message set msg_role='1' where msg_id=?";
66：  PreparedStatement ps = connection.prepareStatement(sql);
67：  ps.setString(1,id);
68：  ps.executeUpdate();
69：  ps.close();
70：  return true;
71：   }
72：}
```

4．allmessages.jsp（数据展示部分）

关键代码　Webroot\admin\allmessages.jsp

```
1：<div id="tanbox">
2：    <ul class="tan">
3：       <li><a href="../admin/MessageServlet?method=ListAll">
4：            全部留言
5：            </a>
6：       </li>
7：       <li><a href="../admin/MessageServlet?method=ListNew">未查看</a></li>
8：    </ul>
9：</div>
10：<!--content-->
11：<div id="right_content">
12：      .....
13：   <div class="part1">
14：      <div class="part_top">
15：         <ul>
16：            <li class="li_1"><a>ID</a></li>
17：            <li class="li_2"><a>选择</a></li>
18：            <li class="li_3"><a>公司名称</a></li>
19：            <li class="li_4"><a>留言时间</a></li>
20：            <li class="li_3"><a>状态</a></li>
21：            <li class="li_5"><a>设置</a></li>
22：         </ul>
23：      </div><!--end of part_top-->
24：      <c:forEach var="List" items="${ListAll}">
25：      <div class="part_content">
26：         <ul>
27：            <li><a>${List.msg_id}</a></li>
28：            <li><a>
29：               <input type="checkbox" name="id" value="${List.msg_id}"/>
30：               </a></li>
31：            <li><a>${List.msg_company}</a></li>
32：            <li><a>${List.issueTime}</a></li>
33：            <c:choose>
34：                  <c:when test="${List.msg_role==0}">
35：                        <li class="li_3"><a style="color:#FF0000">未查看</a></li>
36：                  </c:when>
37：                  <c:otherwise>
38：                      <li><a>已查看</a></li>
39：                   </c:otherwise>
40：                </c:choose>
41：          <li>
42：          <a href="../admin/MessageServlet?method=ShowOne
             &id=${List.msg_id}" target="right">
```

```
43:                查看</a>
44:              <a href="#" onclick="delaff('../admin/MessageServlet?
             method=Del&id=${List.msg_id}')" target="right">删除</a>
45:            </li>
46:              </ul>
47:            </div>
48:            </c:forEach>
49:            <div class="part_bt">
50:              <ul>
51:                <li><a href="#" onclick="selectAll();">[全选] </a>
52:                     <a href="#" onclick="turnOver();"> [反选] </a>
53:                </li>
54:                <li class="li_7"><input type="submit" value="删除选中"/></li>
55:                <li class="li_11">${pages}
56:                </li>
57:              </ul>
58:            </div>
59:          </div>
```

限于篇幅，留言模块中另两个显示页面 mesage_board.jsp 和 message_detail.jsp 的代码不详细列出。

系统其他模块，如管理员信息管理模块、下载表格信息管理模块、友情链接管理模块的处理方法与文章管理模块基本相似，通过前面对文章管理模块的讲解，读者应不难理解这些模块的设计思路，相关代码可直接参考本书所附源码。

第四部分　JSP 速查参考

本部分提供对 JSP 各部分知识的快速参考，包含 JSP 的语言基础、JSP 的页面元素、JSP 的内建对象、JDBC 访问数据库的一般方法，最后还介绍了主流开发工具 MyEclipse 的安装与使用方法。具体章节如下。

第 16 章　JSP 的语言基础
第 17 章　JSP 的页面元素
第 18 章　JSP 的内建对象
第 19 章　JDBC 访问数据库
第 20 章　拾遗补缺

第 16 章　JSP 的语言基础

JSP 是在 HTML 静态网页的基础上，添加了 Java 代码段和特有的标记而形成的动态网页。本章将介绍 HTML、JavaScript，以及 Java 语言的相关知识。

16.1　HTML 语言基础

HTML（HyperText Markup Language，超文本置标语言）是制作静态网页的基本语言，HTML 文件的扩展名为.html 或.htm，它们是可供浏览器解释浏览的文件格式，主要通过一系列的标签（或称标记）来描述页面上多种形式的元素，如图片、超链接和文字等。下面介绍 HTML 4.0 标准中的常用标签。

16.1.1　HTML 的标签和属性

1. HTML 的标签

HTML 页面中有许多由一对尖括号“< >”括起的字母序列，如<html>、<head>、<body>、 等，它们被称为 HTML 的标签。网页上的各种元素分别由不同的标签表示，如<a>标签表示超链接，标签表示图片，<p>标签表示一个段落，<table>标签表示表格。

2. 双标签

标签多数是成对出现的，如<h1>...</h1>，其中<h1>称为开始标签，而</h1>称为结束标签，在开始标签和结束标签中间的内容都受它们的影响。<h1>标签表示以 1 号标题字显示内容。

大部分标签会在页面上呈现出一定的形状或效果，但有一些标签则只起控制或标识作用。如<html>...</html>只是表示浏览器将按 HTML 的语法规范解释这对标签之间的内容。

3. 单标签

有些标签是单个出现的，如图片标签，这种标签应该在右括号前加上“/”，表示标签结束。

4. 属性

许多标签都可通过设定属性值来设计标签的表现细节，每个属性依照“属性名＝"属性值"”的格式进行设定。各属性之间以空格分隔开，属性值可用双引号或单引号括起，如果是双标签，只能在开始标签中设置属性，如“欢迎”，表示文字“欢迎”将以 6 号红色字显示。各属性之间无先后次序，属性是可选的，省略时则采用默认的取值。

16.1.2　HTML 页面的基本结构

一个 HTML 文档由标签<html>和</html>括起，其中又划分为头部和主体部分，其一般结构如下所示：

```
<html>
  <head>
     页面头部
     <title>标题</title>
  </head>
  <body>
     页面主体
  </body>
</html>
```

页面的头部：由标签<head>和</head>括起的部分称为页面的头部，通常用来设定网页的标题、网页内容类型、关键字、网页描述、作者等信息，还可在此引入外部 CSS 文件、外部的 JS 文件等。

页面的主体：由标签<body>和</body>括起的部分为页面的主体，其中可以包含很多标签，如图片标签、<a>超链接标签、<p>文本段落标签等，主体部分的内容则将显示在浏览器的窗口中，也就是我们所浏览到的网页内容。

16.1.3　HTML 的常用标签

HTML 中的标签有很多，这里仅介绍其中最常用的一些，按照其功能进行划分，大致可以分为以下几类。

结构类：<html>、<head>、<title>、<body>。

格式标签：
、 、<marquee>、、、、<pre>。

文本标签：<h1>…<h6>、、<i>、<u>、、。

超链接标签：<a>。

图像标签：。

表格标签：<table>、<tr>、<td>、<td>、<caption>。

框架标签：<frameset>、<frame>、<noframes>、<iframe>。

表单标签：<form>、<input>、<select>、<textarea>、<label>。

头元素标签：<link>、<base>、<meta>、<script>、<style>。

区域标签： 、、 。

以下是几种常用的标签。

1. 头部标签

1）元信息标签<meta/>

在<head></head>内使用，该标签可用于指明网页的类型、作者、关键字、搜索方式等信息。例如，设定网页内容类型及页面编码的字符集：

```
<meta http-equiv="content-Type" content="text/html;charset=utf-8" />
```

该标签告诉浏览器，接收的是一个 HTML 文档，并且网页内容以 UTF-8 编码。

设定网页的关键字：

```
<meta name="keywords" content="塑料,加工" />
```

关键字是提供给搜索引擎使用的，搜索引擎将依据此关键字对网页进行分类。

2）标题标签<title>

在<head></head>内使用，指明网页的标题。格式为“<title>网页标题</title>”。

2. 文档体和段落

以下标签均在<body>和</body>之间使用。

（1）
标签：表示强制换行，该标签没有标签体。

（2） 标签：表示段落标记，用来创建一个段落。格式为“ 段落内容 ”。

（3）<hr>标签：用于在网页中插入一条水平线。格式为“<hr align="对齐方式" size="横线粗细" width="横线长度"

color="横线颜色" noshade>”。Noshade 表示设定线条为平面显示，若默认则有阴影和立体效果。

3. 文字格式标签

（1）标签：该标签用于设置网页中的文字显示的大小、字号和颜色等属性。

格式：

```
<font size="数字" face="字体名" color="颜色">被设置的文字</font>
```

（2）标签：该标签用于将标签内的文字以粗体显示。

格式：

```
<b>需要加粗的文字</b>
```

（3）<i>标签：该标签用于将标签内的文字设置为斜体。

格式：

```
<i>需要显示为斜体的文字</i>
```

（4）<u>标签：该标签用于将标签内的文字设为带下画线。

格式：

```
<u>需要加下画线的文字</u>
```

4. 列表

（1）标签：该标签用于创建一个无序列表，该标签有一个子标签。

格式：

```
<ul type="符号类型">
   <li type="符号类型 1">第 1 个列表项
   <li type="符号类型 2">第 2 个列表项
      … …
</ ul >
```

type 的取值主要有 disc（实心圆点）、circle（空心圆点）和 square（方块）等。

（2）标签：该标签用于创建一个有序列表，该标签有一个子标签。

格式：

```
<ol type="符号类型">
   <li type="符号类型 1">第 1 个列表项
   <li type="符号类型 2">第 2 个列表项
      … …
</ol>
```

type 的取值为 1（数字）、A（大写英文字母）、a（小写英文字母）、I（大写罗马字母）、i（小写罗马字母）。

5. 超链接

<a>标签用于从一个网页跳转到另一个网页，单击超链接即进行跳转动作。

格式：

```
<a href="目标 URL" name="链接名称字符串" target="打开窗口方式">
        浏览器中显示的热点
</a>
```

属性说明如下。

- href：单击后要打开的目标资源（通常即网页）的路径。
- target：表示目标资源的内容将在哪个位置打开，可取值如下。
 - _blank：在新的浏览器窗口中显示目标 URL 中的内容。
 - _parent：将目标 URL 的内容显示在父框架窗口中。
 - _self：在当前窗口中显示目标 URL 中的内容，为默认值。
 - _top：将目标 URL 中的内容显示在没有框架的窗口中。
 - 框架名称：用在框架网页中，用于将目标 URL 中的内容显示在指定的框架中。

也可以通过单击超链接打开系统默认的邮件发送软件，发送电子邮件。

格式：

```
发送电子邮件：<a href="mailto:Email 地址">发送邮件的热点文本</a>
```

6. 表格

该系列的标签可在页面上绘制由行和列组成的表格，并在单元格中显示文字、超链接和图片等其他元素。

（1）<table></table>表示一个表格。

（2）<tr></tr>表示表格中的一行。

（3）<td></td>表示行中的一个单元格，而<td>和</td>之间的内容则显示在该单元格中。

（4）<th></th>表示标题行中的一个单元格，一般为居中加粗显示。

（5）<caption></caption>显示在表格上方或下方的表格标题。

7. 图像

标签用于将一张图片加入到网页中的特定位置。

格式：

```
<img   src="图片文件名"
       alt="简单说明"
       width="图片的宽度"
       height="图片的高度"
       border="边框宽度"
 />
```

8. 表单

（1）<form>标签：该标签用于定义一个表单，在标签之间需要嵌入表单元素，如文本框、列表框、复选框、单选按钮、提交按钮等，通常由<input>标签、<select>标签或<textarea>标签表示。

格式：

```
<form action="资源名称" method="get | post" name="表单名称">
  ……
</form>
```

属性说明如下。

- action：接收本表单数据的资源名称，一般为页面或 Servlet 的 URL。
- method：提交数据的方式，可取 post 或 get 值，默认的取值是 get。

（2）<input>标签：是最常用的输入标签，根据 type 属性的不同值，可以定义出各种不同形态的输入元素。

格式：

```
<input   type=""   name=""   id=""   value="" >
```

属性说明如下。

- name：输入元素的名称，也是发送到服务器的请求参数的名称。
- id：该元素在页面中的唯一标识名，供 CSS 或 JS 使用。CSS 可以通过这个 id 来控制该元素的样式，而 JavaScript 代码也是通过 id 来查找页面上的某个元素的。
- value：初始值，默认值为空字符串。
- type：可取不同的值，在页面上表现为不同的形态，具体取值可参看 5.2.1 小节的表 5-1。

（3）<select>标签与<option>标签：该标签用于创建一个下拉列表框或复选的列表框。

格式：

```
<select size=x name="控件名称" multiple>
   <option selected value="可选择的内容">
</select>
```

属性说明如下。

■ x：指出在带滚动条的选择栏中一次可见的列表项数。

■ multiple：如有该属性则可通过“Ctrl”键与鼠标操作选取多个选项，否则只能单选。

■ selected：表示该项是预置的默认选项（被选中）。

（4）<textarea>标签：该标签用于创建一个允许输入多行文本的文本框。

格式：

```
<textarea name="控件名称" rows="行数" cols="列数">
        多行文本
</textarea>
```

属性说明如下。

■ rows：所显示的文本框的行数。

■ cols：所显示的文本框的列数。

9. 框架标签<frameset>与<frame>

它们可用于在一个浏览器窗口中显示多个相互隔离的网页，所有框架标签需要放在一个总的 HTML 文件中，只记录该框架如何划分，不会显示任何资料。

<frameset>标签用于创建一个框架集，并对框架集中的各帧进行定位。

<frame>标签用于定义具体的帧窗口，该标签必须在<frameset>标签内使用。

格式：

```
<frameset rows="x" cols="y" border="像素数"
        bordercolor="颜色" frameborder="yes | no" framespacing="z">
    <frame src="URL" name="框架名" border="像素数" bordercolor="颜色"
        scrolling="yes|no|auto|noresize" />
    <frame src="URL"/>
</frameset>
```

属性说明如下。

■ rows：用来规定主文档中有几行帧窗口及各个窗口的大小，该属性值可以由多个百分数、绝对像素值或星号（*）组成，各个值之间以逗号分隔。

■ cols：用来规定主文档中有几列帧窗口及各个帧窗口的大小，该属性值可以由多个百分数、绝对像素值或星号（*）组成，各个值之间以逗号分隔。

■ noresize：如有则表示不允许各窗口改变大小，默认设置是允许各窗口改变大小。

举例：

```
<html>
    <head>… … </head>
      <frameset rows="80,*" cols="*" border="0" framespacing="0">
          <frame src="…" name="topFrame">
          <frame src="…" name="bottomFrame">
       </frameset>
</html>
```

16.2　JavaScript 简介

JavaScript（简称 JS）是由 Netscape 公司推出的客户端脚本语言，JavaScript 代码是由浏览器来解释执行的，目前基本上所有的浏览器都内嵌了对 JavaScript 的支持。JS 主要应用于制作网页特效、客户端数据验证等。

16.2.1　JavaScript 的基本语法

JS 的语法与 Java 有很多相似之处，但也存在一些差别，通过比较二者语法风格上的异同，可尽快掌握 JavaScript 的基本语法。

- 与 Java 一样，JS 是对大小写敏感的。
- JS 的分支语句、循环语句格式与 Java 基本相同。
- JS 中的语句不必以分号结尾，但为了与 Java 保持一致，建议加上分号。
- JS 中的字符串可用双引号括起，也可用单引号括起。
- JS 中的变量定义采用“var s;”这种形式，即不指出该变量的类型，其类型由所赋的值决定。一个变量甚至可以不定义就直接使用。
- JS 中的函数定义以关键字 function 引导，不指出参数及返回值的类型。

16.2.2　在页面中加入 JS 代码

可以采用以下 3 种形式在页面中加入 JS 代码。

（1）直接写在 HTML 标签的事件处理属性中。

例如，当输入框得到输入焦点时，将框内的文字清空：

```
<input type="text" value="请在此输入用户名" onfocus="this.value=" " />
```

（2）在页面头部的<script>...</script>标签之间编写 JS 代码，格式如下：

```
<script>
    JS 代码
</script>
```

（3）将 JS 代码编写成独立的 JS 文件，在<head></head>部分利用<script>标签导入。

格式：

```
<script src="JS 文件名">
</script>
```

16.2.3　JS 的事件处理

JS 通过对用户事件的响应完成对事务的处理。JS 的事件很多，除了定时器事件外，主要是针对用户的键盘输入或鼠标动作而进行的处理，如 mouseover（鼠标移上）事件、mouseout（鼠标移开）事件、focus（输入元素得到焦点）事件、blur（输入元素失去焦点）事件、submit（提交表单）事件等。可通过 onXXXX 的形式为相应的事件指定处理代码，其中 XXXX 为事件名称。

1. 为元素指定事件处理代码

格式：

```
onXXX=""
```

例：

```
<input type="text" name="msg" id="msg" onblur="print()" />
```

通常还需为相关的元素指定 id，以便查找控制这个元素，该 id 在整个页面中应是唯一的。

2. 定义函数

格式：

```
function functionName(param1,param2,...){

}
```

例：

```
function print(){ alert(document.getElementById("msg"));}
```

此处使用了 document 对象的 getElementById(id)方法，可根据 id 找到相应的元素。

16.2.4　表单数据验证的一般方式

当用户填写了表单并单击“提交”按钮进行提交时，则发生提交事件。通过对表单的提交事件进行处理，可以实现数据的客户端验证。这时表单的一般形式为：

```
<form action="next.jsp" onsubmit="return check()">
    ...
    <input type="submit" value="submit" />
</form>
```

具体的验证工作则在函数 check()中进行。仅当 check()函数的返回值为 true 时，表单内容才会继续提交到<form>标签的 action 属性所指定的页面，否则不会提交。

16.3　Java 语言基础

16.3.1　Java 的类和对象

Java 程序设计是以类为基础的。可以说，Java 程序设计的过程就是定义和应用类的过程。

所谓类，是对一类事物的特征和行为的概括。例如，游戏中的玩家，应该描述其用户名、等级、经验值、装备等信息，还应该具有建房、练剑、战斗等各种技能。当现实世界中的某人参与到此游戏中成为玩家后，程序就按照事先设计好的“玩家”类，将该玩家的各项数据具体化，同时该玩家可以具有各项技能，因此，这个玩家是一个具体化的例子，在面向对象的程序设计中，称之为“玩家”类的一个实例，也称为对象。当然，一个游戏中除了“玩家”这类事物外，还有诸如“场景”、“敌人”等其他类型的事物。

1. 类的定义

类中包含两类成员：属性和方法。属性描述事物的特征，而方法则描述事件所具有的行为或能力。定义一个类，就是定义其中的属性和方法。

2. 类定义的格式

```
[类修饰符]class 类名 [extends 父类名称]
{
属性列表;
方法定义;
}
```

其中方法名与类名相同的方法称为构造方法，该方法无返回值，当创建该类的对象时被系统自动调用。

```
public class Player {
   private String name;
   public void setName(String name){ this.name=name;}
   public void print(){System.out.println(name);}
}
```

3. 类的实例化

当设计好一个类后，就可以用它作为“模板”，创建一个实例来使用它了。一般通过 new 关键字来创建一个实例。例：

```
Player p1=new Player();
```

创建了实例后，类中定义的公有成员可以通过实例名加上圆点操作符来访问或调用，如：

```
p1.setName("黑风骑士");
p1.print();
```

若一个方法是静态方法，即定义时使用了关键字 static，则可以直接通过类名来调用它。

16.3.2 常用的 Java 类

这里简单介绍使用频率较高的几个类和接口，其他类或接口的资料请参考 Java API 说明文档。

1. java.lang.String 类

封装了对字符串的一般处理。

常用方法如下。

- boolean equals(String s)：比较两个字符串是否相同，返回 true 或 false，如 s1.equals("he")，比较 s1 的内容是否为"he"。
- boolean equalsIgnoreCase(String s)：忽略大小写，比较两个字符串是否相同。
- int length()：返回字符串的长度。
- int indexOf(String str)：返回子串 str 在字符串中出现的索引位置，第 1 个字符的索引序号为 0。
- String substring(int start,int end)：截取当前字符串从索引 start 到 end-1 处的子串，如“String s1="How are you"; String s2=s1.substring(4,7);”，则 s2 为"are"。

- String trim()：删除字符串两端的空格并返回。
- static String valueOf(double d)、valueOf(int i)、valueOf(float f)、valueOf(long l)：返回数值所对应的字符串形式，均为静态方法，如表达式 String.valueOf(125)的值为字符串 125。

2. java.util.Date 类与 java.text.SimpleDateFormat 类

- 得到系统的当前日期：Date now=new Date();。
- 将日期转换为指定格式的字符串：SimpleDateFormat 类可将日期转换为指定格式的字符串。在实例化一个 SimpleDate Format 对象时应指明采用的时间模式字符串，常用模式字母的含义如表 16-1 所示。

表 16-1　表示日期格式的模式字母表

符　　号	含　　义
y	年
M	月
d	日
h	时（12 小时制，1—12）
H	时（24 小时制，0—23）
m	分
s	秒
S	毫秒
a	上午/下午 标记符（am/pm）

3. java.util.ArrayList 类

List 接口的实现类，可称之为“可变长”数组。

常用方法如下。

- void add(E o)：在列表的末端插入元素 o。
- void add(int index,E o)：在指定的位置插入元素 o。
- void clear()：清除其中的所有元素。
- E get(int index)：返回指定位置的元素。
- int indexOf(E o)：寻找元素 o 在列表中第一次出现的位置。
- boolean isEmpty()：列表是否为空。
- E remove(int index)：删除列表中指定位置的元素，所有后继（右侧）的元素向左移动，返回被删除的元素。
- int size()：返回列表中的元素个数。

16.3.3　Java 的异常处理机制

在程序的执行过程中可能会发生各种异常，如将字符串转换为整数时发生数字格式异常，访问数据库时连接失败而发生异常等。Java 语言中的 try 语句用于捕捉程序中发生的异常，然后可根据情况进行具体的处理，或再次抛出这个异常。

try 语句的语法格式：

```
try{
```

```
//代码处理段，可能发生异常
}
catch(异常类  e )
{
//异常处理代码
}
finally
{
//最终执行段，无论是否发生异常，均执行这段代码
}
```

当 try 后面大括号内的代码段中有某条代码发生异常时，则中止括号内其他代码的执行，转去执行 catch 后面的处理代码。

根据情况，可以有 finally 段的代码，也可以没有。finally 代码段是无论 try{}中的语句块是否发生异常，最终都要执行的一段代码。

第 17 章　JSP 的页面元素

除 HTML 格式的页面代码外，JSP 页面文件主要由注释、动作、声明、表达式和代码段（Scriptlets）等元素组成，如表 17-1 所示。

表 17-1　JSP 中的页面元素

符　　号	所表示的页面元素	示例及说明
<%--　--%>	注释	例：<%-- 文件功能：验证登录--%> 此注释内容不会被传送到客户端
<%　%>	代码段	例：<% if (u1.check()) ...　%> 可在其中编写 Java 代码段
<%!　%>	代码段	例：<%! int count=0; %> 其中只能定义变量或方法
<%@　%>	指令	例： <%@page contentType="text/html;charset=utf-8" %> 或 <%@ include file="head.jsp"%> 指示 JSP 引擎如何翻译本页面
<jsp:　/> 或 <jsp:　> </jsp:　>	动作	例：<jsp:forward　page="login.jsp"　/> JSP 的标准动作
<%=　%>	表达式	<%=uname%>

需要注意的是，除了表达式可以与其他元素结合使用外，其他语法元素一般是不能嵌套组合的。例如，在 JSP 的代码段标记“<%”和“%>”之间，不能出现 JSP 的动作或指令，因此下面的写法是错误的：

```
<%
if ( u1.check())　{
  <jsp:forward page="index.jsp" />
}
%>
```

17.1　JSP 的注释

JSP 是以 HTML 为基础的，而其中又可能含有 Java 代码段，因此根据注释所处的代码段的语言环境，可分别采用 HTML 格式的注释或 Java 格式的注释。除此之外，JSP 还提供了自身风格的注释：

```
<%--这是 jsp 独有的注释--%>
```

这种注释和写在 Java 代码段中的注释一样，不会传递到客户端。

17.2　JSP 的指令

一条指令无论写在页面的什么位置，都对整个页面有效。

JSP 的指令用于从 JSP 发送一个消息到 JSP 引擎上，它们本身一般不会向客户产生任何输出，即不产生可视化的输出内容，只是用来指示 JSP 引擎在进行 JSP 页面的翻译工作时，提供全局信息，帮助指导如何处理页面上的其他代码。

JSP 中的指令都是用符号“<%@”和“%>”括起的，主要有以下 3 种：

- page 指令；
- include 指令；
- taglib 指令。

指令元素在使用时采取以下的格式：

```
<%@ 指令名称 属性="值" 属性="值" … %>
```

17.2.1　page（页面）指令

page 指令是 JSP 页面中使用频率最多的指令。通常我们会把它写在 JSP 页面的顶部。

page 指令用来定义影响到整个页面的重要属性，如网页内容的类型、网页中字符的编码方式、网页需导入的包、发生异常时的处理页面、使用的编程语言和缓冲区的大小等信息。

它的语法格式为：

```
<%@ page 属性名 1="属性值 1" ... 属性名 n="属性值 n" %>
```

page 指令一共有 13 个属性可以设置，说明如下。

- language="java"：指定页面所使用的脚本语言，目前只能是 Java。
- import="包名.类名|包名.*"。
- contentType="页面 MIME 类型[;charset=字符集名称]"。
- pageEncoding="字符集"。
- extends="包名.类名"。
- session="true|false"：是否可使用 session，默认为 true。
- buffer="none|8KB|sizeKB"：页面所使用的缓冲区的大小，默认值与服务器有关，至少是 8KB，如果为 none 则不缓存。
- autoFlush="true|false"：设置缓冲区满时是否自动输出，默认为 true。
- isThreadSafe="true|false"：是否支持多线程，默认为 true。
- info="文本信息"。
- errorPage="异常处理页面的 url"。
- isErrorPage="true|false"：是否为错误处理页面，默认为 false。
- isELIgnored="true|false"：是否允许使用 EL（表达式语言），默认为由 web.xml 描述文件的版本确定。

书写时请注意区分大小写。上面语法格式中的中括号表示括起的内容是可选的，而符号“|”则表示“或者”的意思。其中最常用的属性请参考表 17-2。

表 17-2　page 指令的主要属性

属　性	说　明	示　例
pageEncoding	指定 JSP 页面的字符编码	pageEncoding="utf-8"
contentType	指定输出内容的 MIME 类型及字符集	contentType="text/html;charset=gb2312"
import	导入 Java API 或自定义的类包，该属性可以在一个页面中多次指定	import="java.sql.*" import="com.sky.Goods"
errorPage	指定一个 JSP 页面，本页面抛出的所有异常均交由该页面处理	errorPage="err.jsp"
isErrorPage	指定本页面是否可作为其余 JSP 页面的异常处理页面，默认值是 false	isErrorPage="true"

可以在一条 page 指令中指定多个属性，也可以分开多条指定。但除 import 属性可以在一个页面中多次出现外，其余属性只能指定一次，不能重复出现。

17.2.2　include（包含）指令

include 指令所实现的文件包含往往又称为“静态包含”，它用于通知 JSP 容器将指定的资源内容内嵌到当前页面中该条指令出现的位置。这里的资源可以是 JSP 文件、HTML 文件，或者是其他类型的文本文件。

语法格式：

```
<%@ include file="URL"/>
```

其中的 URL 表示被包含的资源的文件路径（含文件名）。

注意：

- 被包含的页面若出现了设置页面编码的 page 指令，则主页面中所出现的 page 指令的设置必须与此完全相同，否则可能引起翻译时的错误。
- file 属性的值不能是变量。

17.2.3　taglib（标签库）指令

标签库指令是与开发人员自定义的标签库配套使用的，这个指令允许使用者自定义标签。设计者首先要开发标签库（设计标签类），编写.tld 配置文件，然后使用 taglib 指令告诉 JSP 容器怎样找到标签描述文件和标签库，以及在 JSP 页面引用这个标签时的前缀。

taglib 指令的语法格式：

```
<%@ taglib uri="标签库的 URI" prefix="标签的前缀" %>
```

例如：

```
<%@ taglib uri="myLib" prefix="cc" %>
```

不能使用 jsp、jspx、java、javax、servlet、sun、sunw 等作为前缀。

17.3　JSP 的代码段

JSP 程序中进行逻辑判断或其他处理的代码片段，需要使用 Java 语言编写。为了告诉 JSP

容器这段代码是有别于 HTML 代码的，必须使用特殊的符号"<%"和"%>"括起。

例如：

```
<%
out.println("hello");
%>
```

JSP 页面中还可声明 Servlet 的成员方法或成员变量，该 Servlet 由 JSP 文件经引擎转换得到，此时需用“<%!”和“%>”括起，如：

```
<%!
public int add(int a,int b)
{
    return a+b;
}
%>
```

之后可以在本页面的代码段中调用此方法。

在“<%!”和“%>”之间只能出现方法或变量的定义，不能出现其他语句。

17.4　JSP 的表达式

JSP 中的表达式可以看成是一种简单的输出形式，使用符号“<%=”和“%>”括起，作用是计算并输出该表达式的值。该表达式在页面中的位置，也就是其计算结果所显示的位置。注意“<%=”这 3 个字符之间不能有空格，且符号“%>”前面没有分号。

例如：

```
<%= a+b %>
```

将输出变量 a 与变量 b 相加的结果。

17.5　JSP 的标准动作

JSP 的动作是遵循 xml 语法规范的，而 JSP 的标准动作都是以 jsp:开头的。JSP 的标准动作在处理请求时起作用，影响到 JSP 运行时的行为和发送给客户端的应答。

下面只列出最常用的几个动作。

17.5.1　<jsp:param>动作

该动作必须配合<jsp:include>动作、<jsp:forward>动作或<jsp:plugin>一起使用，作用是向指定的网页传送参数。

语法格式：

```
<jsp:param name="name" value="value"/>
```

其中，name 为属性的名称，value 为该属性的值。

17.5.2　<jsp:include>动作

该动作用于包含一个静态或动态的文件。若包含的是静态文件，则直接将内容包含进来，若是动态文件，则先执行该文件，然后将执行结果包含进来。

语法格式：

```
<jsp:include page="URL"/>
```

或

```
<jsp:include page="URL">
    <jsp:param name="name" value="value" />
</jsp:include>
```

与 include 指令不同，这里 page 属性的取值可以是表达式的形式。

17.5.3　<jsp:forward>动作

该动作直接在服务器端将用户的请求转发到另一个页面处理，与 response 对象所实现的重定向不同的是，这种转发直接在服务器端内部进行。

语法格式：

```
<jsp:forward page="URL" />
```

或

```
<jsp:forward page="URL">
   <jsp:param name="name" value="value" />
</jsp:forward>
```

17.5.4　使用 JavaBean 的系列动作

JavaBean 被称为 JSP 中的组件，其本质是一个 Java 类，但是必须符合一定的设计规范。

<jsp:useBean>动作用于在页面中创建或引用一个 JavaBean 实例，<jsp:setProperty>动作和<jsp:getProperty>动作则用于设置或读取 JavaBean 实例的属性值。

1. <jsp:useBean>动作

语法格式：

```
<jsp:useBean class="package.class"   id="id"   scope="page|request|session|application"/>
```

或

```
<jsp:useBean class="package.class"   id="id"   scope="page|request|session|application">
    ...
</jsp:useBean>
```

该动作主要指明所要创建或引用的 JavaBean 实例的包名、类名、id、有效范围等。

属性说明如下。

■ class：指出 JavaBean 类的包括包名在内的完整类名。

■ id：JavaBean 实例的名称。

■ scope：该 JavaBean 实例的有效范围，可以有 4 种取值，分别为 page、request、session 和 application，默认值为 page。具体含义如下。

- page：该 Bean 实例仅在本页面范围内有效。
- request：该 Bean 实例在请求所延续的范围内有效。
- session：该 Bean 实例在会话范围内有效，即可以在属于同一次会话的页面中使用。更确切地说，是可以被同一个客户端在该应用的所有页面中使用。
- application：该 Bean 实例可在整个应用范围内使用，即可以被所有客户端在本应用的任意页面中共享使用。

2. <jsp:setProperty>动作

设置某个已创建的 JavaBean 实例的属性值，该动作实际上是调用 JavaBean 中的 set 方法去实现的，因此应保证所设置的属性具有 set 方法。该动作可以有以下几种格式。

1）直接指定属性的值

```
<jsp:setProperty  name="beanName"  property="propertyName"
  value="propertyValue" />
```

指定 JavaBean 实例的某个属性值，该实例的名称由 beanName 指定，属性名称由 propertyName 指定，属性值由 propertyValue 指定。

2）由请求参数指定属性值

```
<jsp:setProperty  name="beanName"  property="propertyName"
param="parameterName" />
```

指定 JavaBean 实例的某个属性值，该实例的名称由 beanName 指定，属性名称由 propertyName 指定，属性值为请求参数 parameterName 的值。

3）同名请求参数自动传递值

```
<jsp:setProperty  name="beanName"  property="*" />
```

指定 JavaBean 实例的属性值，若名称为 beanName 的 JavaBean 实例，其属性名称与请求参数的名称相同，则自动将其值设为同名的请求参数的值。

3. <jsp:getProperty>动作

返回某个已创建的 JavaBean 实例的某个属性的值。

语法格式：

```
<jsp:getProperty name="beanName" property="propertyName" />
```

属性说明如下。

■ name：JavaBean 实例的名称。

■ property：属性的名称。

第 18 章　JSP 的内建对象

JSP 为简化页面的开发提供了一组内置的对象，即在 JSP 页面中无须创建就可直接使用的对象。它们提供了网络编程中常用的一些基本功能，如输出信息，获取来自客户端的信息，向客户端发送控制信息，在客户端与服务器的会话期间存取信息等。这些内建对象不需要 JSP 的开发人员进行实例化，而由 JSP 容器进行管理，在所有的 JSP 页面中都能使用内置对象。

JSP 中共有 9 个内建对象，如表 18-1 所示。

表 18-1　JSP 的内置对象

对 象 名 称	类　　型	含　　义
application	javax.servlet.ServletContext	代表本 JSP 页面所属的应用本身
config	javax.servlet.ServletConfig	代表本 JSP 页面的配置信息
exception	java.lang.Throwable	封装了其他页面中的异常和错误
out	javax.servlet.jsp.JspWriter 或 java.io.printWriter	代表本 JSP 页面的输出流，用于向浏览器输出内容
page		代表页面本身
pageContext	javax.servlet.jsp.PageContext	代表 JSP 的页面上下文
request	javax.servlet.http.HttpServletRequest	封装了来自客户端的请求信息
response	javax.servlet.http.HttpServletResponse	封装了服务器对客户端的响应
session	javax.servlet.http.HttpSession	代表客户端与服务器的一次会话

一般的 JSP 页面可使用除 exception 外的其余 8 个内置对象，而 exception 对象仅在声明为错误处理的 JSP 页面中才可使用，这类页面需使用 page 指令将 isErrorPage 属性值设为 true。

18.1　out 对象

out 对象用来向客户端输出数据。默认情况下，网页的输出内容并不是直接输出到客户端的，而是先写到一个输出缓冲区中。out 对象代表本页面的输出流。其常用方法如下。

- void print（Object o）：向缓冲区中写入数据，参数可以是基本数据类型或其他引用类型。
- void println（Object o）：向缓冲区中写入数据，并换行。
- void clear()：清除缓冲区的内容，但不把数据输出到客户端。
- void clearBuffer()：清除缓冲区的内容，并把数据输出到客户端。
- void close()：关闭输出流，清除所有内容。
- void flush()：立即输出缓冲区中的数据。
- int getBufferSize()：返回缓冲区的大小（单位为 KB）。
- int getRemaining()：返回缓冲区中剩余空间的大小（单位为 KB）。

18.2　request 对象

request 对象是 JSP 程序设计中最常使用的内置对象，使用 request 对象可以对客户端在发出请求时传送给服务器的信息进行访问。其常用方法如下。

- void setCharacterEncoding(String code)：设置请求信息编码时所采用的字符集。当请求中的信息含有中文信息时，通常需要设置请求的编码，一般可设为 GB2312、GBK 或 UTF-8。建议设为 UTF-8。
- String getCharacterEncoding()：返回请求的字符编码方式。
- String getParameter(String name)：返回客户端发送的请求参数的值。
- request 对象的 getParameter 方法返回的数据均为 String 类型，若输入的是数值型数据，则提取后需要进行类型转换。

例如：

```
String uname=request.getParameter("username");
int age=Integer.parseInt(request.getParameter("username"));
```

- String [] getParameterValues(String name)：有时，从客户端传送给服务器页面的一个参数可能包含多个数值，通常可能是在表单中采用了复选框来收集数据的时候。
- String getRemoteAddr()：返回客户端的 IP 地址。
- void setAttribute(String name, Object obj)：向 request 对象中写入属性，参数 name 表示属性的名称，而 obj 表示属性的值。
- Object getAttribute(String name)：返回存储在 request 对象中的名称为 name 的属性值，取出后通常需进行强制类型转换。若不存在该名称的属性，则返回空值。
- void removeAttribute(String name)：删除存储在 request 对象中由 name 指定的属性。
- String getQueryString()：以字符串的形式返回以 get 方式传送的所有参数名称及值。
- Cookie [] getCookies()：返回本网站保存在客户端的所有 Cookie 对象，结果是一个 Cookie 类型的数组。

18.3　response 对象

response 对象封装了从服务器返回给客户端的所有信息。response 对象的常用方法如下。

- void sendRedirect(String location)：该方法可将客户端的请求重新定向到由参数 location 指定的位置。
- void addHeader(String name,String value)：添加 HTTP 响应头，如果同名的 header 存在，则会覆盖原来的值。
- void setHeader(String name,String value)：设置由 name 指定的 HTTP 的文件标头的值。
 - 指定网页的失效时间：

```
response.setHeader("expires", "时间数");
```

 - 页面定时重定向：

```
response.setHeader("refresh","秒数;url=重定向网页的 URL ");
```

- void addCookie(Cookie c)：添加一个 Cookie 对象，当响应发送到客户端时，该 Cookie 信息通常会被保存在客户端的硬盘上。保存在客户端的 Cookie 信息可以通过 request 对象的 getCookies 方法提取出来。
- void setContentType(String contentType)：通常用于动态地改变当前网页的 MIME 类型。MIME（Multipurpose Internet Mail Extensions，多用途因特网邮件扩展）用于指明网页的类型，使用“类型/子类型”的形式指定。常见的 MIME 类型如表 18-2 所示。

表 18-2　MIME 的类型

MIME 类型	说　明
application/msword	Microsoft Word 文档
application/pdf	Acrobat PDF 文件
applicaiton/x-msexcel	Microsoft Excel 文件
audio/x-wav	WAV 格式的音频文件
text/html	HTML 格式文本文档
text/css	CSS 样式表文件
text/plain	普通文本文档
image/jpeg	JPEG 格式图片文件
image/gif	GIF 格式图片文件
video/mpeg	MPEG 格式视频文件

它的默认值为“text/plain”，因此一般 HTML 格式的文档需显式地指明为“text/html”类型。

- void sendError(int sc)：向客户端传送 HTTP 错误状态码，如 500 表示服务器内部错误，404 表示所访问的资源不存在或已经失效。

18.4　session 对象

session 对象用于在客户端与服务器的一个会话期间保持客户端的信息。其常用方法如下。

- void setAttribute(String name,Object value)：向 session 中写入一个属性，该属性的名称为 name，值为 value。若同名的属性已经存在，则将覆盖原来的值。
- Object getAttribute(String name)：返回名称为 name 的属性的值，返回值为 Object 类型，通常需根据具体应用进行强制类型转换。若不存在指定名称的属性，则返回 null。
- void　removeAttribute(String name)：删除 session 中由 name 指定的名称的属性。
- void setMaxInactiveInterval(int interval)：设置所允许的最长的非活动时间，有人形象地称之为“发呆时间”，单位为秒。Tomcat 默认为 30 分钟。
- void invalidate()：销毁当前 session 对象。

18.5　application 对象

application 对象用于保存在应用范围内的共享数据，这些数据可被所有访问该应用的客户端共享。服务器启动且自动创建 application 对象以后，只要没有关闭服务器，该对象将一直存在。其常用方法如下。

- void setAttribute(String name,Object value)：向 application 中存入一个属性，名称由 name 指定，值由 value 指定。
- Object getAttribute(String name)：获取 application 中由 name 指定的名称的属性的值，若不存在，则返回 null。
- void removeAttribute(String name)：移除 application 中由 name 指定的属性。
- String getInitParameter(String name)：获取应用的初始参数值，该参数的名称由 name 指定。可以在网站的配置文件 web.xml 中设置网站的初始参数。
- String getRealPath(String path)：获取与虚拟路径对应的物理存储路径。

18.6　config 对象

config 对象表示 Servlet 的配置信息，当一个 Servlet 程序初始化时，JSP 引擎把某些消息通过该对象传递给这个 Servlet，此消息包括 Servlet 程序初始化时所需要的参数及服务器的有关信息。config 对象是实现了 javax.servlet.ServletConfig 接口的类的实例，其常用方法如下。

- ServletContext getServletContext()：调用这个方法可以返回一个含有服务器相关信息的 ServletContext 对象，即 JSP 内置对象 application。
- String getInitParameter(String name)：调用这个方法可以返回 Servlet 程序初始参数的值，参数名由 name 指定。
- Enumeration getInitParameterNames()：调用这个方法可以返回一个枚举对象，该对象由 Servlet 程序初始化所需要的所有参数的名称构成。这些参数可以在配置文件 web.xml 中设置。

18.7　exception 对象

exception 对象是 Throwable 的实例，代表运行时的异常信息，只有在错误处理页面（使用 page 指令将属性 isErrorPage 设为 true 的页面）中才能使用。

通过 exception 对象，可以捕获到其他页面转发过来的出错信息，常用方法如下。

- String getMessage()：返回详细的异常信息字符串。
- String toString()：返回此异常的简短信息描述。
- printStackTrace(PrintStream s)：将此异常及其调用栈轨迹输出到指定的输出流。

18.8　page 对象

page 对象代表的是 JSP 实现类的实例，即它就是 JSP 本身。它代表 JSP 实现类对象的一个

句柄，只有在 JSP 页面的范围内才是合法的，在开发的过程中很少使用 page 对象。

18.9　pageContext 对象

pageContext 对象是 JSP 中很重要的一个内置对象，通过该对象的方法可以访问其他 8 个 JSP 内部对象，还可以直接访问绑定在 application 对象、page 对象、request 对象、session 对象上的 Java 对象。pageContext 对象相当于 JSP 程序中所有对象功能的集成者。该对象的创建和初始化通常对 JSP 程序员是透明的，在一般的 JSP 程序中使用较少，这里不做详细介绍。

第 19 章　JDBC 访问数据库

19.1　一 般 步 骤

JSP 程序访问数据库的方法就是 Java 程序访问数据库的方法，在 Java 程序中通过 JDBC 操作数据库的一般步骤如下。

（1）在项目中添加 JDBC 驱动包。

可在项目中添加对驱动 jar 文件的引用，或者将目标数据库产品的 JDBC 驱动复制到 Tomcat 安装目录的 lib 子目录或项目的 WEB-INF\lib 目录下。这些驱动通常可在数据库厂商的官方网站下载。

（2）实例化并注册数据库驱动类。

根据数据库的类型注册相应的驱动类，各主流数据库注册驱动的代码如下。

■ Oracle 8/8i/9i 数据库（thin 模式）：

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

■ SQL Server 2000 数据库：

```
Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");
```

■ SQL Server 2005 数据库：

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
```

■ MySQL 数据库：

```
Class.forName("com.mysql.jdbc.Driver");
```

（3）建立数据库连接，取得 Connection 对象。

常见的几种数据库产品的连接方式如下。

```
String user="账号";
String password="密码";
```

■ Oracle 8/8i/9i 数据库（thin 模式）：

```
String url="jdbc:oracle:thin:@localhost:1521:实例名";
Connection conn="DriverManager.getConnection(url,user,password);
```

■ SQL Server 2000 数据库：

```
String url="jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=数据库名";
Connection conn="DriverManager.getConnection(url,user,password);
```

■ SQL Server 2005 数据库：

```
String url="jdbc: sqlserver://localhost:1433;DatabaseName=数据库名";
String user="账号";
```

```
String password="";
Connection conn="DriverManager.getConnection(url,user,password);
```

■ MySQL 数据库：

```
String  url="jdbc:mysql://localhost:3306/ 数 据 库 名 ?user= 账 号 &password= 密 码 &useUnicode=true&characterEncoding=utf8";
Connection conn=DriverManager.getConnection(url);
```

（4）建立 Statement 对象或 PreparedStatement 对象。

通过 Connection 对象创建 Statement 对象或 PreparedStatement 对象。

例如：

```
//建立 Statement 对象
Statement stmt=conn.createStatement();
//建立 PreparedStatement 对象
String sql="select * from users where userName=? and password=?";
PreparedStatement pstmt=conn.prepareStatement(sql);
pstmt.setString(1,"admin");
pstmt.setString(2,"4321");
```

（5）执行 SQL 语句。

例如：

```
//执行静态 SQL 查询
String sql="select * from users";
ResultSet rs=stmt.executeQuery(sql);
//执行动态 SQL 查询
ResultSet rs = pstmt.executeQuery();
//执行 insert、update、delete 等语句，应先构建相应的 SQL 语句
stmt.executeUpdate(sql);    或    pstmt.executeUpdate();
```

（6）访问结果集 ResultSet 对象。

例如：

```
while(rs.next()) {
   out.println("你的第一个字段内容为: " + rs.getString(1));
   out.println("你的第二个字段内容为: " + rs.getString(2));
}
```

（7）关闭各资源对象，释放所占用的资源。

例如：

```
rs.close();
stmt.close();
pstmt.close();
con.close();
```

19.2　常用的接口或类

下面介绍通过 JDBC 访问数据库时几个主要类或接口的常用方法，其中的几个接口均具有关闭资源的方法，格式为 void close()。下文不再一一指出。

19.2.1　DriverManager 类

管理一组 JDBC 驱动程序的基本服务类。

getConncetion 方法常用的重载形式有以下两种。

形式一：

```
static Connection getConnection(String url)
```

试图建立到给定的数据库 URL 的连接。

形式二：

```
static Connection getConnection(String url,String user,String password)
```

试图用给定的用户名和密码创建到给定的数据库 URL 的连接。

19.2.2　Connection 接口

■ createStatement 方法：该方法可创建 Statement 的对象，该方法有以下两种重载形式。

形式一：

```
Statement createStatement()
```

形式二：

```
Statement createStatement(int resultSetType, int resultSetConcurrency)
```

第二种形式中，createStatement 方法带有两个参数，分别表示游标类型与上锁类型。

■ prepareStatement 方法：返回 PreparedStatement 接口实现类的对象。

格式：

```
PreparedStatement prepareStatement(String sql)
```

■ prepareCall 方法：返回 CallableStatement 接口实现类的对象，利用该对象可调用存储过程。

格式：

```
CallableStatement prepareCall("{call <precedure name>>[(<arg1>,<arg2>, ...)]}")
```

19.2.3　Statement 接口

Statement 接口主要提供了用于执行 SQL 语句的方法。

■ executeQuery 方法：用于执行 select 语句，执行完毕后，返回一个 ResultSet 类型的对象，该对象封装了从数据库服务器返回的查询结果。

格式：

```
ResultSet executeQuery(String sql)
```

该结果集有一个游标，游标的类型由创建 Statement 对象时的参数决定。

■ executeUpdate 方法：executeUpdate 方法用于执行 insert、update、delete 或 DDL 语句，返回值为受影响的记录的行数。

格式：

```
int executeUpdate(String sql)
```

■ execute 方法：执行给定的 SQL 语句，该语句可能返回多个结果。当第一个结果为 ResultSet 类型时，返回 true，若第一个结果为受影响的行数，或者不存在任何结果，则返回 false。

这个方法使用起来较为麻烦，一般可用 executeQuery 或 executeUpdate 方法替代。

格式：

```
boolean execute(String sql)
```

19.2.4　PreparedStatement 接口

PreparedStatement 是 Statement 的子接口，称为预编译的 Statement，用于反复执行相似的 SQL 语句，有利于提高执行的效率。所执行的 SQL 语句中一般带有参数。

除具有 Statement 接口的所有方法外，PreparedStatement 接口还有以下方法。

■ executeQuery 方法：用于执行 select 语句，执行完毕后，返回一个 ResultSet 类型的对象，该对象封装了从数据库服务器返回的查询结果。

格式：

```
ResultSet executeQuery()
```

■ executeUpdate 方法：executeUpdate 方法用于执行 insert、update、delete 或 DDL 语句，返回值为受影响的记录的行数。

格式：

```
int executeUpdate()
```

■ setXXX 方法的格式如下：

```
void setXXX(int index,Object o)
```

方法名称中的 XXX 需根据表中对应列的数据类型而定，如数据类型为 String 时，该方法名为 setString，参数 index 为占位符在 SQL 语句中出现的序号，参数 o 为该占位符指定的值。

19.2.5　CallableStatement 接口

CallableStatement 接口用于执行数据库的存储过程，继承了 PreparedStatement 接口。在执行存储过程之前，必须注册所有 OUT 参数的类型；它们的值是在执行后通过此接口提供的 getXXX 方法检索的。

除具有 PreparedStatement 接口的方法外，常用的方法如下。

■ registerOutParameter 方法：注册输出参数的类型。

格式：

```
void registerOutParameter(int index,String type)
```

■ getXXX 方法：读取存储过程的输出结果。

格式：

```
XXX getXXX(int index)
```

方法名中的 XXX 及返回值类型均需根据存储过程的输出参数的类型而变化。

19.2.6　ResultSet 接口

表示查询数据库的结果集合，通过执行查询数据库的 SQL 语句生成。ResultSet 对象具有指向其当前数据行的游标。最初，游标被置于第一行之前。next 方法将指针移动到下一行。

■ boolean next()：将游标移向结果集的下一条记录。

■ boolean previous()：将游标移向结果集的前一条记录。

■ boolean first()：将游标移向结果集的第一条记录。

■ boolean last()：将游标移向结果集的最后一条记录。

■ boolean absolute(int rows)：将游标移向指定的行号。

■ boolean relative(int rows)：以当前位置为参照，将游标移动指定的行数。

■ boolean isFirst()：判断游标是否位于第一条记录。

■ boolean isLast()：判断游标是否位于最后一条记录。

■ getXXX(int columnIndex)或 getXXX(String columnName)：返回结果集中当前行的指定列的值，方法名中的 XXX 由该列的数据类型决定。参数可以是该列在结果集中的序号或列名。

第 20 章　拾遗补缺

20.1　关系数据库与 SQL 回顾

20.1.1　关系数据库的基本结构

关系数据库是以二维表的形式组织数据的，一个数据库可以由多张表组成。表的一行为一条记录。

如表 20-1 所示表示某书店的图书信息表，表中 ISBN、书名、作者、出版社、出版时间、单价称为字段（或称为列），表中的一行表示某图书的信息，称为一条记录，每一条记录都应该提供上述 6 个字段的值。在所有的字段中，能唯一标识这条记录的字段称为主键，如上表的 ISBN，所出版的图书都有自己唯一的一个 ISBN 编号。

表名、字段名一般不建议使用中文名称，通常可采用相应的英文或中文拼音的形式。如上表的 6 个字段可分别用字段名 bisbn、bname、bwriter、bpublisher、btime、bprice，表名可命名为 book。

表 20-1　图书信息表

ISBN	书　　名	作　　者	出 版 社	出 版 时 间	单　　价
7-111-12345	数据库基础	张军	天外	2008-9-1	23.00
7-5327-2569-3	挪威的森林	村上春树	上海译文	2001-2-1	18.8
.....					
978-7-03-022522-1	Java Web 开发技术详解	汪大乐	北京科海	2008-8-1	63.00

20.1.2　SQL 语言

SQL（Structured Query Language）语言是操作关系数据库的通用语言，它提供了对数据库进行数据定义、数据操纵（增、删、改）、数据查询、数据控制（权限管理）等多种功能。Java 程序采用 JDBC 访问数据库时，也是通过执行特定的 SQL 语句来完成对数据库的操作的，因此有必要掌握基本的 SQL 语言。这里我们只简单地复习一下在 Web 应用中使用频率最高的 SQL 语句，即建表语句及记录的增、删、查、改语句。

1. 基本语法

SQL 语言是不区分大小写的。SQL 语句中的字符串常量、日期型常量应以单引号括起，而其他类型的常量则无须用单引号括起。

下面的语法格式中，[]表示所括起的部分是可选的，大括号表示其中的内容可以重复多次。

2. 表的创建

语法格式：

```
create table <表名> (
```

```
<列名> <数据类型> [列级约束]
{,<列名> <数据类型> [列级约束]...}
[,表级约束]
)
```

3. 插入

该语句可以将记录插入到表中，即在表中添加新的记录。

语法格式：

```
insert into <表名> [(列名列表)] values (值列表)
```

4. 删除

删除表中所有符合条件的记录。若未指明条件，则默认为删除表中的所有记录。

语法格式：

```
delete from <表名> [where <条件表达式>]
```

5. 修改

修改表中符合条件的记录的字段值。

语法格式：

```
update <表名> set <列名>=<表达式> [, ... n] [where <条件表达式>]
```

6. 查询

查询语句按照指定的条件查询指定的一张或多张表，并得到符合条件的所有记录的集合。Java 程序从数据库中读取数据时应该使用查询语句。

语法格式：

```
select <目标列名或表达式序列> from <数据源> [where <查询条件>] [group by <分组依据>][having <组提取条件>] [order by <排序依据列> asc|desc]
```

20.2　MyEclipse 的安装与使用

MyEclipse 可选择“PLUG-IN”版作为 Eclipse 的插件使用，也可选择“ALL IN ONE”版独立安装使用。这里介绍在 Eclipse 中安装 MyEclipse 插件的方法。

Eclipse 是目前开发 Java 应用程序的主流 IDE 工具，它是基于 Java 的可扩展开发平台的，其核心是提供软件开发工具集成服务的 Eclipse 基础平台，在此基础上通过插件来构建各种开发环境。例如，通过安装不同的插件，Eclipse 可用于开发 Java、C/C++、PHP 等多种应用程序。Eclipse 本身附带有一个标准的 Java 开发工具插件，可以用来开发最基础的 Java 应用程序，但是如果需要方便快捷地开发 JSP、Servlet 等程序，还需要安装如 MyEclipse 这样的插件。

20.2.1　Eclipse 及 MyEclipse 的安装

不同版本的 Eclipse 需配合安装不同版本的 MyEclipse 插件，这里选择的是 Eclipse 3.3，应配合 MyEclipse 6.0 使用。

1. 安装 Eclipse 3.3

Eclipse 是免费的开源软件，可以登录网站 http://download.eclipse.org 进行下载，得到一个压缩包 eclipse-SDK-3.3.1.1-win32.zip。只需将这个压缩包解压到磁盘上即完成了安装。这里将其解压到 D:\Program Files 目录下，解压后的文件夹内容如图 20-1 所示。

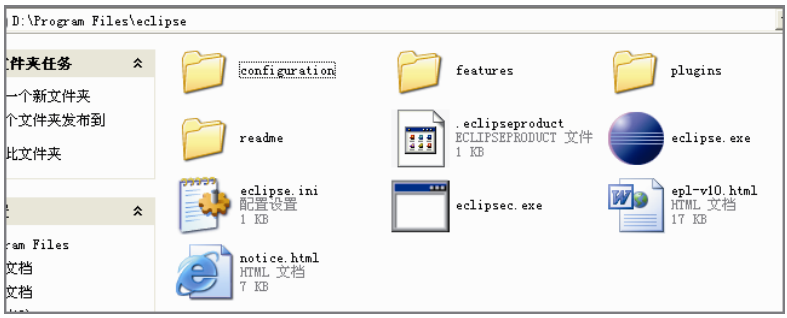


图 20-1　Eclipse 解压后的文件夹内容

单击其中的 eclipse.exe 文件即可启动 eclipse，启动时的画面如图 20-2 所示。

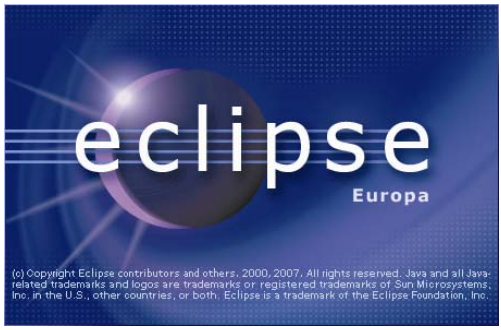


图 20-2　eclipse 的启动画面

第一次启动 eclipse 时，会弹出如图 20-3 所示的对话框，我们在此设定工作空间的路径，以后项目文件将默认存放在此目录之下。例如，将工作空间路径设置为 D:\eprog，则以后新建的一个项目 proj1 的文件将自动存放在 D:\eprog\proj1 目录下。

注意将复选框“Use this as the default and do not ask again”勾上，以后均按此设置而不再询问。

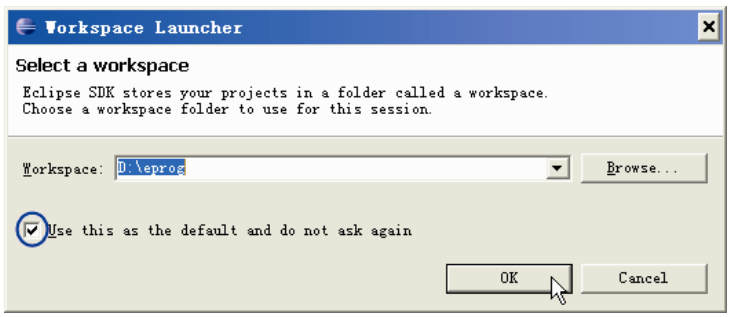


图 20-3　设置程序的默认存放位置

在对话框中单击“OK”按钮后，出现如图 20-4 所示的欢迎界面。



图 20-4　Eclipse 的欢迎界面

由于我们开发的是 Web 应用程序，还需安装 MyEclipse，因此先退出 Eclipse。在下图的对话框中选上“Always exit without prompt”，以后退出时就不会再弹出这个询问窗口了。

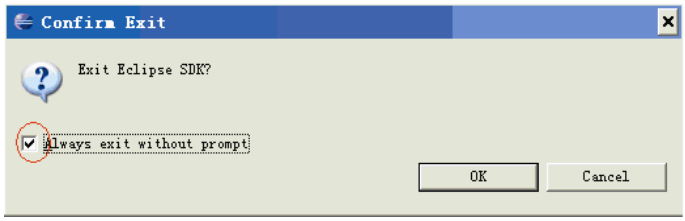


图 20-5　退出时的设置

2. 安装 MyEclipse 6.0 插件

MyEclipse（MyEclipse Enterprise Workbench）是由 Genuitec 公司开发的商业版插件，简称 MyEclipse，可以免费使用一个月。它的官方网站是 http://www.myeclipseide.com，在主页中单击 Download 即可进行下载。配合 Eclipse 3.3 使用的是 MyEclipse 6.0 版。

同一版本的 MyEclipse 有两种不同的安装版本可以下载使用，一种是“ALL IN ONE”即完整版，就是将 Eclipse 和 MyEclipse 打包在一起的版本；另一个是“PLUG-IN”，即插件版。由于前面已经安装了 Eclipse，因此只需要下载 MyEclipse 的插件版就可以了。

双击安装文件 MyEclipse_6.0.1GA_E3.3.1_Installer.exe，开始安装 MyEclipse 6.0，安装文件首先开始解压工作，如图 20-6 所示。解压完成后，将出现安装过程介绍（Introduction）窗口，在此窗口中单击“Next”按钮。

将弹出如图 20-7 所示的对话窗口，在此选择接受协议，然后单击“Next”按钮继续。

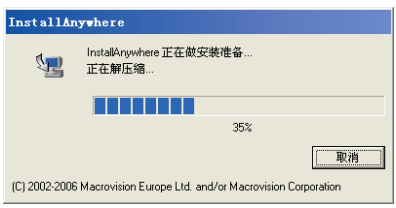


图 20-6　安装包正在解压

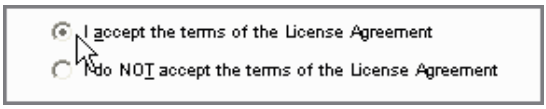


图 20-7　接受协议

接下来在图 20-8 所示的对话窗口中输入之前解压好的 Eclipse 的路径，这里必须输入正确的路径，否则安装无法继续。输入完毕后单击“Next”按钮继续。

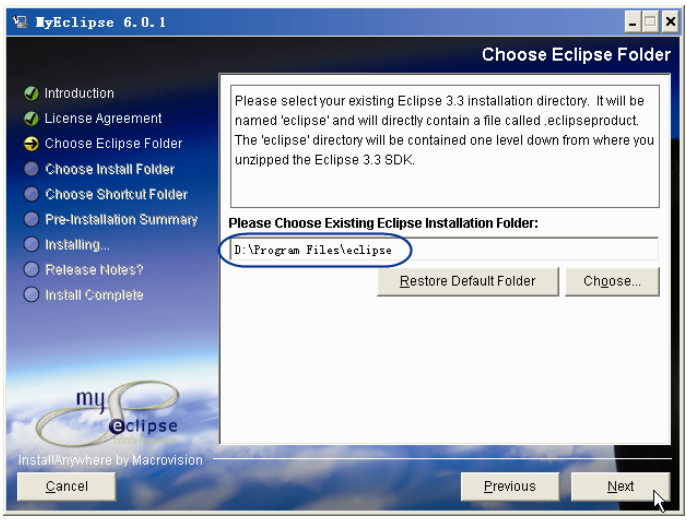


图 20-8　指出 Eclipse 的安装目录

在图 20-9 所示的对话窗口中，选择 MyEclipse 的安装路径，可以自由指定，但是请注意，路径中不要使用中文。

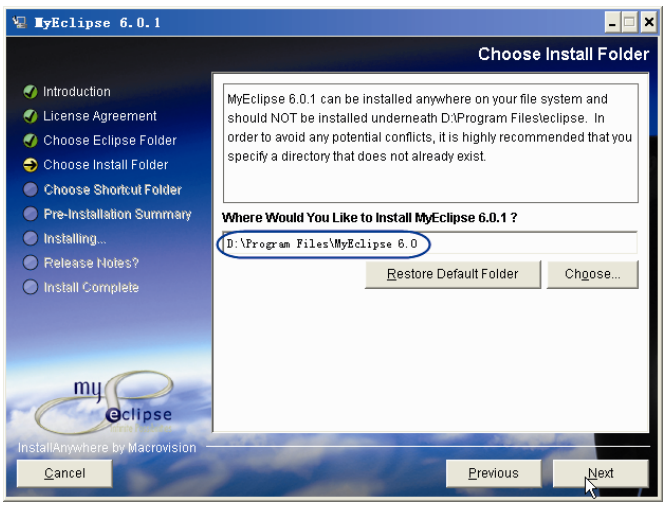


图 20-9　设置 MyEclipse 的安装路径

接下来的对话窗口是让我们选择需要生成哪种形式的快捷图标，保持默认选择的即可，单击“Next”按钮继续。

此时出现的安装前信息总览（Pre-Install Summary）窗口显示了前面所有的设置信息，如果发现设置不正确，可以单击“Previous”按钮，返回前一步重新设置；如果全部正确，则可以单击“Install”按钮正式开始安装。

安装程序将开始文件复制等工作，如图 20-10 所示。这个过程将持续几分钟，请等到下一个画面出现。

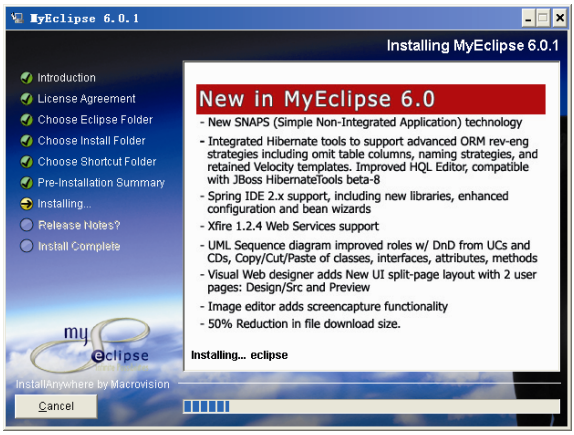


图 20-10　MyEclipse 正在安装的界面

当完成了安装复制等工作后，将出现询问是否查看说明文件的窗口，可以打开说明文件进行阅读。若不想阅读，则如图 20-11 所示选择“No”即可，单击“Next”按钮进入下一步。

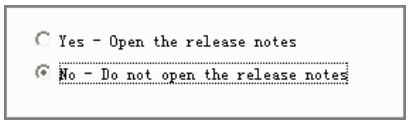


图 20-11　不打开说明文档

在图 20-12 所示的窗口中单击“Done”按钮。至此，MyEclipse 的安装工作全部结束。

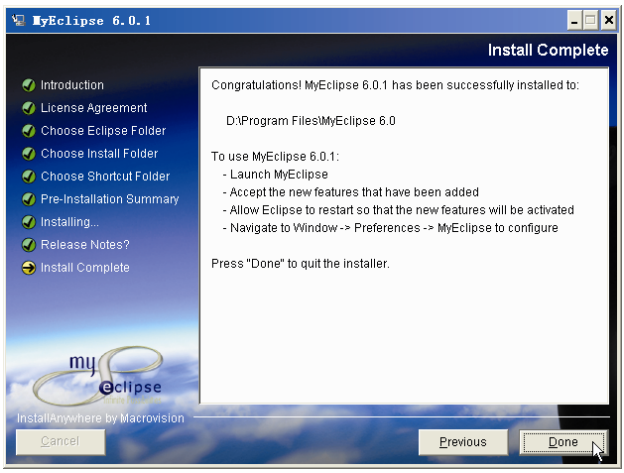


图 20-12　MyEclipse 安装完成

3. 注册 MyEclipse

可通过双击 Eclipse 安装目录下的 eclipse.exe 来启动 MyEclipse，也可通过单击程序组 MyEclipse 6.0 中的 MyEclipse 选项来启动，启动画面如图 20-13 所示。

由于 MyEclipse 是商业软件，因此若超出了 30 天的试用期还未输入注册码的话，会出现如图 20-14 所示的警告框，有可能不能再继续使用。若你已经购买了注册码，则可以单击图中的“Enter Subscription”按钮，在图 20-15 所示的窗口中输入注册信息，之后单击“Finish”按

钮完成输入。



图 20-13　MyEclipse 启动界面

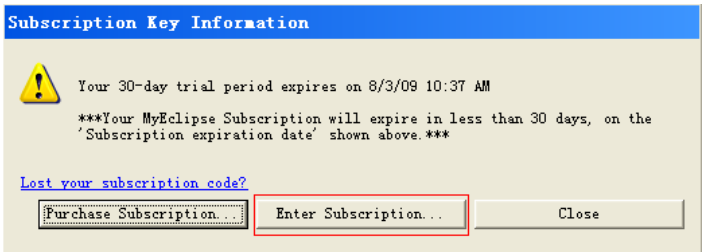


图 20-14　软件注册提示

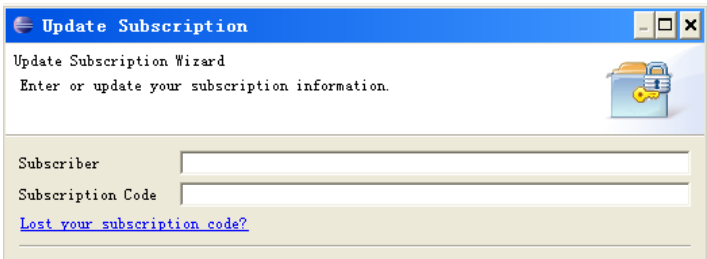


图 20-15　注册码输入界面

当然，也可以在主菜单的“MyEclipse”项中选择“Subscription Information”来输入注册信息。

20.2.2　MyEclipse 的使用

安装好 Eclipse 及插件 MyEclipse 后，就可以使用它来开发我们的 JSP 程序了。

启动 MyEclipse，首先出现的是欢迎界面，其中包含了一些学习教程等，可以先关闭欢迎窗口，以后如果还需要打开，在主菜单的“Help”项中选择“Welcome”即可。

1. 认识 Eclipse 的使用界面

当打开一个项目后，Eclipse+MyEclipse 的常规使用界面如图 20-16 所示。下面我们一起来认识一下这个开发环境。



图 20-16　安装 MyEclipse 插件后的 Eclipse 使用界面

可以看到，界面上打开了多个窗口，也称为视图（view），Eclipse 中把同时打开的一组窗口的集合称为透视图（perspective）。在完成不同的任务或处于不同的状态时，可能需要打开一组不同的视图。例如，开发 Java 基础应用时，打开的是 Java 透视图；安装了 MyEclipse 插件后，默认打开的是 MyEclipse Java Enterprise 透视图；而在调试时，打开的是 Debug，即调试透视图。使用不同的透视图是为了方便完成不同类型或不同阶段的任务，好比生活中做不同的事情时需要不同的一组工具一样。IDE 的右上角显示了目前所使用的透视图的名称，如图 20-17 所示，可以单击名称左边的图标或右边的箭头按钮切换到不同的透视图。

界面最上方是主菜单，主菜单下面是工具栏。在 MyEclipse Java Enterprise 透视图中还打开了多个窗口，每个窗口的右上角都有最小化和最大化按钮，可以单击最小化按钮将一些窗口最小化，以给代码编辑窗口较大的显示空间。

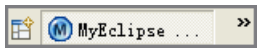


图 20-17　IDE 右上角的透视图切换选项

下面结合项目的开发流程来逐一认识这些窗口。

2. 创建新项目

首先新建一个 Web 项目。单击工具栏最左侧的新建按钮或它右侧的下拉按钮，弹出项目类型选择窗口，选择其中的“Web Project”，如图 20-18 所示。

在图 20-19 所示的对话窗口中输入项目名称，其余按默认设置，然后单击“Finish”按钮完成。

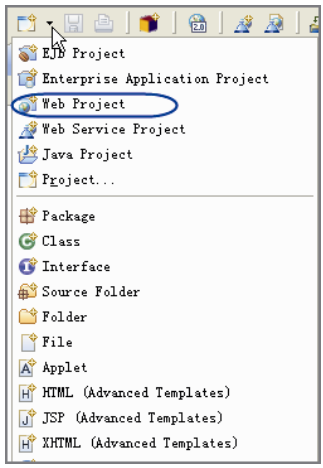


图 20-18　选择新建项目的类型

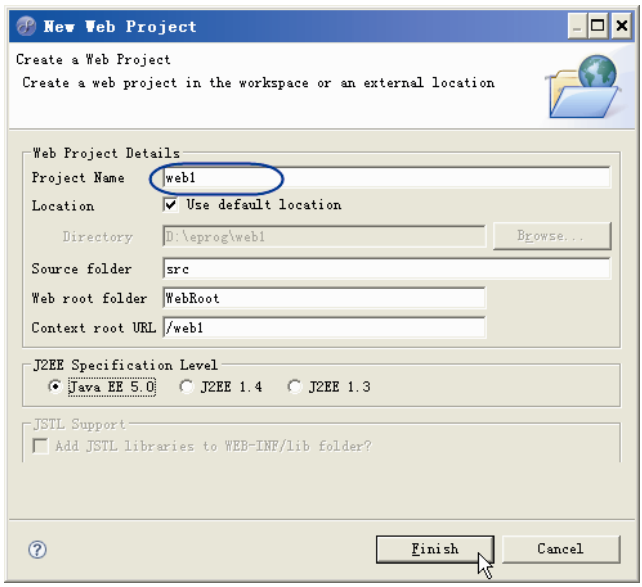


图 20-19　输入项目名称

此时项目的所有文件都存放在默认的工作空间目录下，而网站的访问路径（即图中的 Context root URL）也默认以项目名称为准。

在左侧的 Package Explorer（包资源管理）窗口中，可以看到已经创建好的项目框架，并且已经自动添加了一个默认的主页面 index.jsp，如图 20-20 所示。

我们在第一次打开 Eclipse 时已经将工作空间的默认路径设为 D:\eprog，因此项目 web1 的实际存放位置为 D:\eprog\web1，所有 JSP 或其他页面文件则默认存放在其中的 WebRoot 子目录下，而 src 子目录将用于存放.java 源文件。

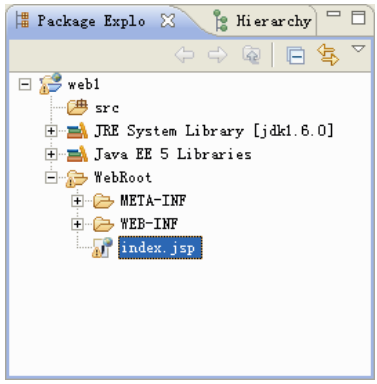


图 20-20　项目的包资源管理窗口

3. 编辑代码

在包资源浏览窗口（Package Explorer）中，双击文件名 index.jsp，该文件的内容将出现在代码编辑窗口中，如图 20-21 所示。

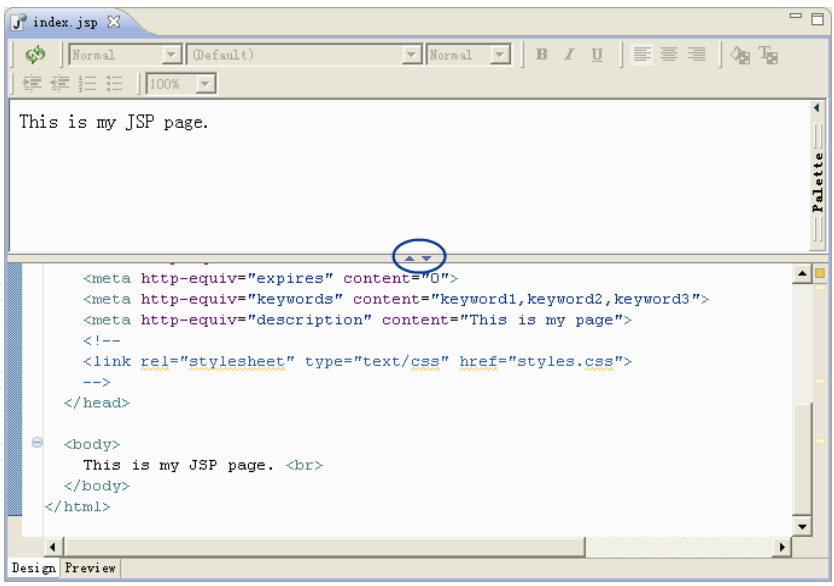


图 20-21　代码编辑窗口

代码编辑窗口上方是页面效果设计窗口，如果不需要，可以单击上下两个窗口之间的边框上的向上箭头，将代码窗口最大化。

在代码的<body>标签后面输入一些文字，可以看到编辑窗口上方的文件名前面多了个“*”号，这表示该文件已被修改，但未保存，如图 20-22 所示。可以单击工具栏上的按钮进行保存。



图 20-22　文件修改后未保存时的文件名

屏幕右侧的大纲视图则显示了文件 index.jsp 的文档结构，如图 20-23 所示。

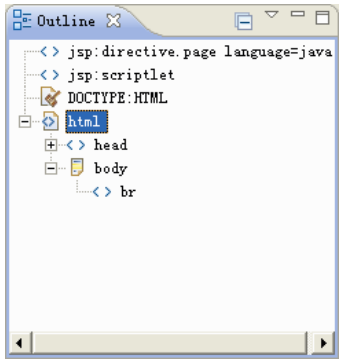


图 20-23　大纲视图

4. 配置 Tomcat

要测试页面的运行情况，需要将项目发布到服务器上。MyEclipse 自带了一个 Tomcat 服务器，这里介绍为 MyEclipse 配置一个外部的 Tomcat 的方法。

首先应安装一个不依赖于任何开发工具的 Tomcat 服务器，安装方法可参考第 11 章的相关内容。

Tomcat 安装完毕后，在 MyEclipse 的主菜单“Windows”中选择“Preferences”，将进入 MyEclipse 的配置选项窗口。在此窗口中依次选择“MyEclipse”→“Servers”→“Tomcat”→“Tomcat 6.x”，将打开如图 20-24 所示的对话窗口。

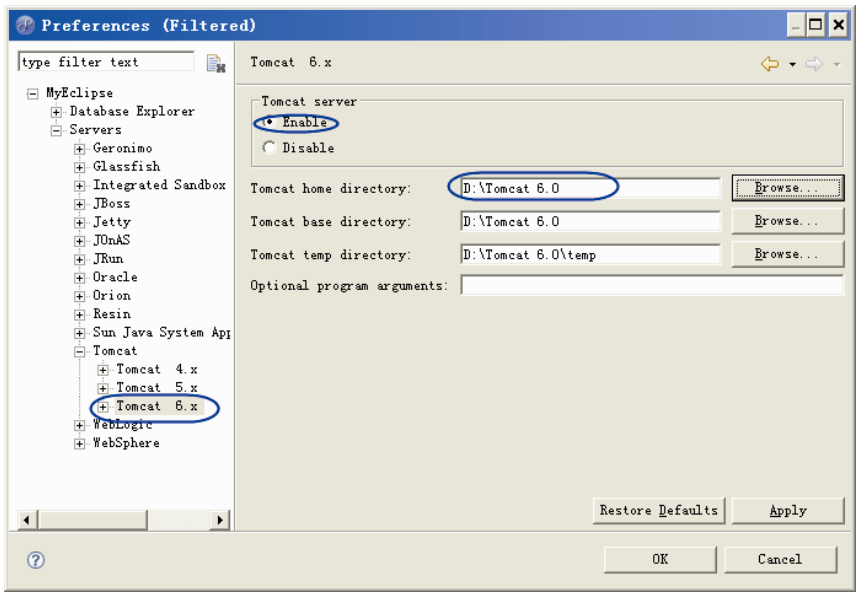


图 20-24　配置 Tomcat

在窗口中将“Enable”选项选中，并在“Tomcat home directory”输入框中输入所安装的 Tomcat 的根路径，单击“OK”按钮，配置完成。

5. 运行 JSP 页面

现在我们来测试一下页面 index.jsp 的运行情况，需要先发布项目，工具栏上的一组按钮 将帮助我们轻松实现。基本步骤是发布项目→启动服务器→打开浏览器→输入网址

进行测试。具体步骤如下。

（1）项目的发布。

单击工具栏上的“发布”按钮，出现图 20-25 所示的窗口。

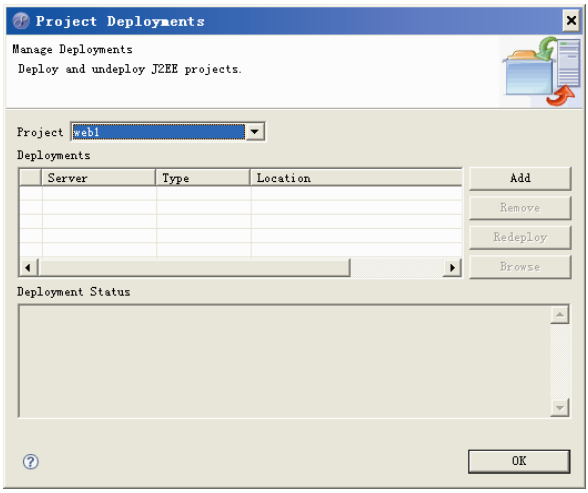


图 20-25　项目发布

目前需要发布的项目是 web1，因此“Project”输入框的内容不变。单击右侧的“Add”按钮，在图 20-26 所示的窗口中选择 Web 服务器，这里选择的是 MyEclipse 自带的“Tomcat 6.x”，若以后独立安装了一个外部的 Tomcat，也可选择其他服务器。

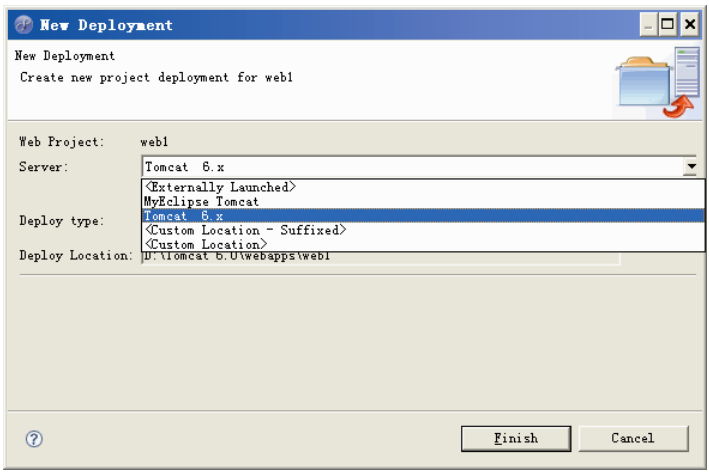


图 20-26　选择服务器

单击窗口下方的“Finish”按钮完成操作，如发布成功则得到如图 20-27 所示的界面。

在这个窗口中还可以删除以前的发布结果（Remove 按钮）、重新发布（Redeploy 按钮）。当修改了项目中的程序后，需要重新发布才能得到正确的结果。

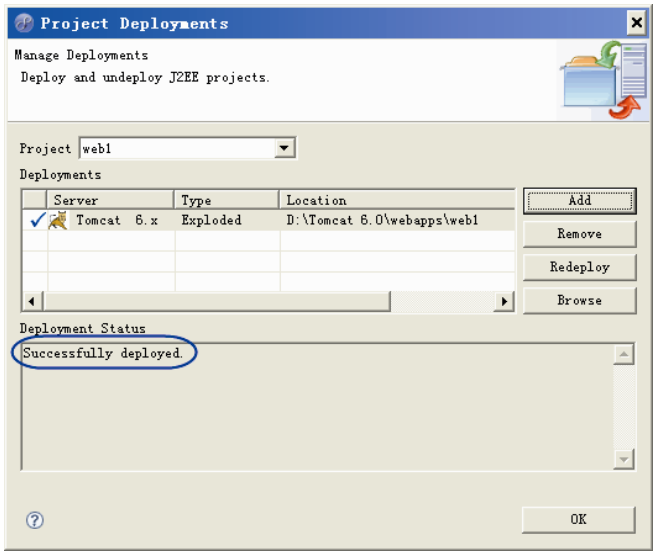


图 20-27　项目发布成功

（2）按工具栏上的旁边的向下箭头按钮，选择要启动的服务器，单击“Start”启动。注意，这里应选择与之前发布时相同的服务器，如图 20-28 所示。

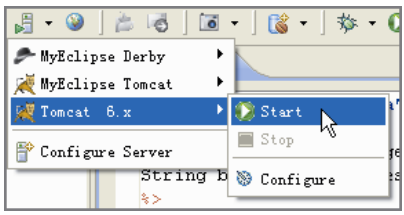


图 20-28　启动服务器

在下方的输出视图中可以看到启动服务器的信息，如图 2-29 所示。

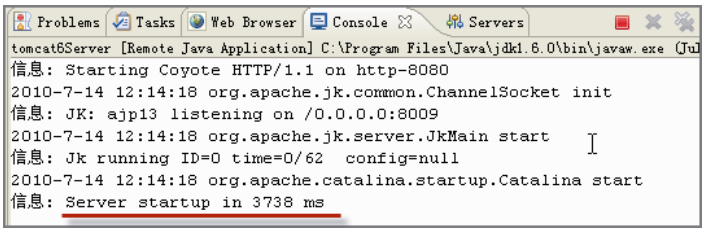


图 20-29　服务器正常启动时的控制台信息

控制台窗口中最后显示的信息“Server startup in 3738 ms”表示服务器已经在所示的时间内启动成功。

（3）浏览页面。

完成上述工作后，就可以浏览一个 JSP 页面的运行结果了。单击工具栏上的按钮，启动 MyEclipse 中内置的浏览器窗口，也可直接启动外部的浏览器。在地址栏中输入网址 http://localhost:8080/web1/index.jsp，将在浏览器中得到如图 20-30 所示的运行结果。

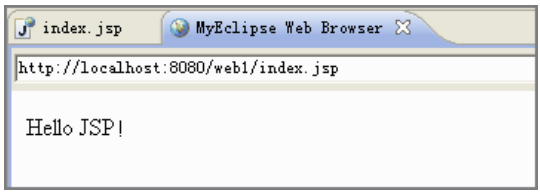


图 20-30　浏览页面的运行结果

在浏览器的地址栏中显示的地址 http://localhost:8080/web1/index.jsp 中，其中的 8080 是 Tomcat 服务器默认的运行端口，web1 是项目名称。由于 index.jsp 是项目的默认主页，因此也可以省略该文件名而直接输入 http://localhost:8080/web1。

6. 添加 JSP 页面

先在包资源浏览管理（Package Explorer）窗口中单击项目名称，然后单击工具栏上最左边“新建”按钮旁边的向下箭头，在文件类型选择窗口中选择“JSP”，如图 20-31 所示。

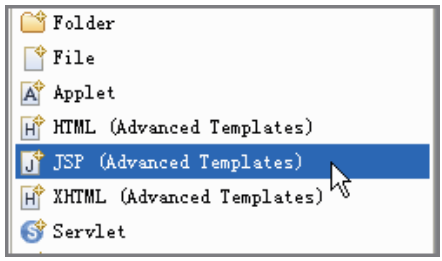


图 20-31　选择文件类型

在图 20-32 所示的窗口中输入文件名称，单击“Finish”按钮完成。

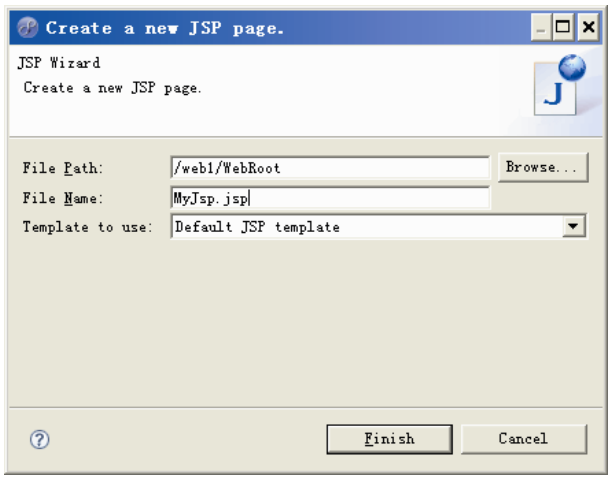


图 20-32　设置文件名

在代码编辑窗口输入 JSP 代码，然后使用与前面相同的方法测试该页面的运行结果。

为了方便查找代码，可以将行号显示出来。用鼠标右击编辑框左侧的灰色边条，在弹出的右键菜单中选择“Show Line Numbers”，如图 20-33 所示。



图 20-33　显示行号的设置方法

有时编辑后代码排列很不整齐，此时可以对代码进行格式化。在代码编辑窗口单击鼠标右键，在弹出的菜单中选择“Format”即可，如图 20-34 所示。

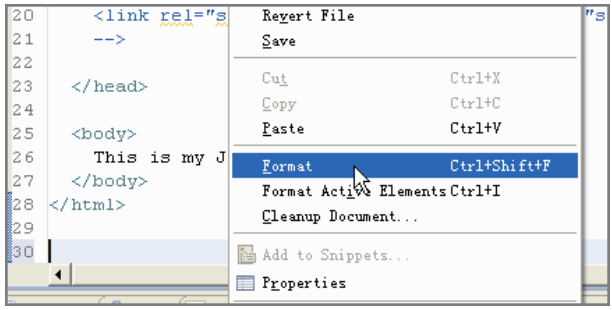


图 20-34　格式化代码

7. 程序的调试

当我们在浏览器地址栏中输入网址去访问一个在其中设置了断点的 JSP 页面时，会自动进入调试状态。因此要调试一个程序，应在其中先设置断点，程序中任意一行可执行的代码都能成为有效的断点。

1）设置断点

双击行号左边的灰色边框，可以在某行设置断点，再次双击，则取消断点。断点以小球图案标识，如图 20-35 的第 29 行所示。

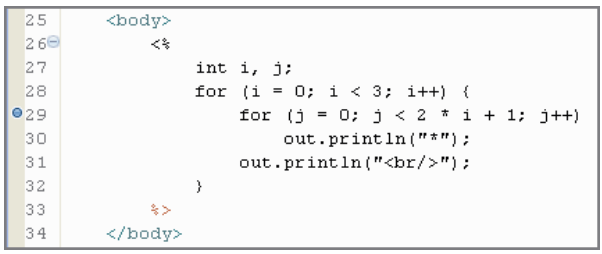


图 20-35　断点的设置

2）调试透视图

当启动调试时，将自动切换到调试透视图，系统会弹出如图 20-36 所示的提示框，询问是否打开调试透视图，选择“Yes”即可。若希望以后不再询问，则可将“Remember my decision”勾选上。

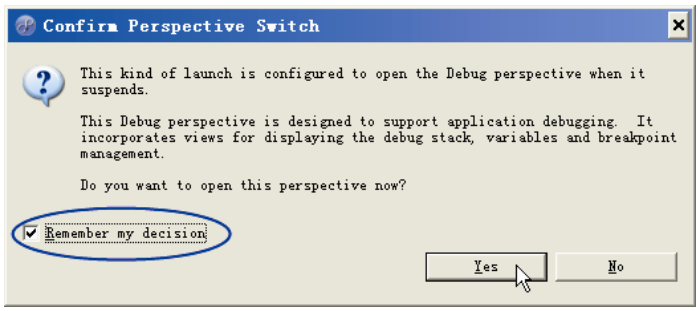


图 20-36　设置默认的调试透视图的询问框

此时，将切换成调试透视图界面。

调试窗口上方出现了一组调试用的按钮，单击其中的按钮可逐行或逐段执行代码，如图 20-37 所示。



图 20-37　调试工具按钮

可按功能键进行调试，按功能键“F6”可以执行一行代码，按功能键“F5”则可跟踪到代码所调用的方法内部。代码窗口显示了目前的执行情况，行号前有一个小箭头的那行代码是即将被执行的代码，同时该行代码被以不同的背景色标识，在图 20-38 中是第 31 行。

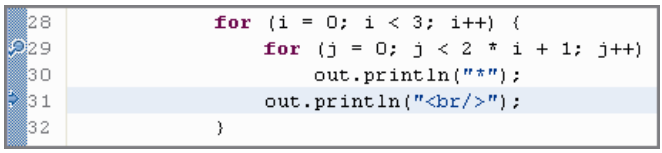


图 20-38　即将被执行的代码

界面上的“Variables”窗口中，显示了页面中所有变量的当前值，如图 20-39 所示。在逐条或逐段执行代码时，观察这些变量的值是否正确，往往有助于判断出程序中的逻辑错误。

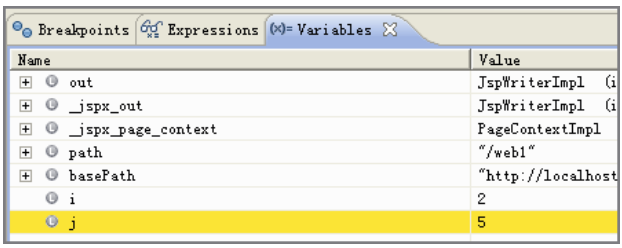


图 20-39　变量查看视图

“Breakpoints”窗口中可以查看程序中的断点设置，而“Expressions”窗口中可观察感兴趣的表达式的值。

单击调试按钮组中的红色“停止”按钮可以结束调试状态，按窗口右上方

中的“>>”按钮可切换回原来的 MyEclipse Java Enterprise 透视图。

20.3　参 考 链 接

Eclipse 官方网站：http://www.eclipse.org/。

MyEclipse 官方网站：http://www.myeclipseide.com。